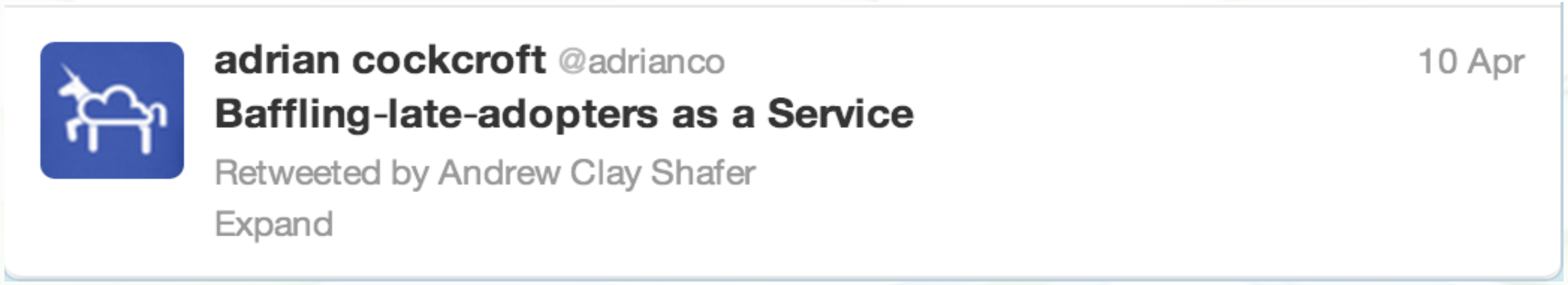
BV Battery Ventures

# Speed and Scale: How to get there.

Adrian Cockcroft @adrianco May 2014



adrian cockcroft @adrianco 10 Apr

Baffling-late-adopters as a Service

Retweeted by Andrew Clay Shafer

Expand

## Typical reactions to my Netflix talks…

“You guys are crazy! Can’t believe it”
– 2009

“What Netflix is doing won’t work”
– 2010

It only works for ‘Unicorns’ like Netflix”
– 2011

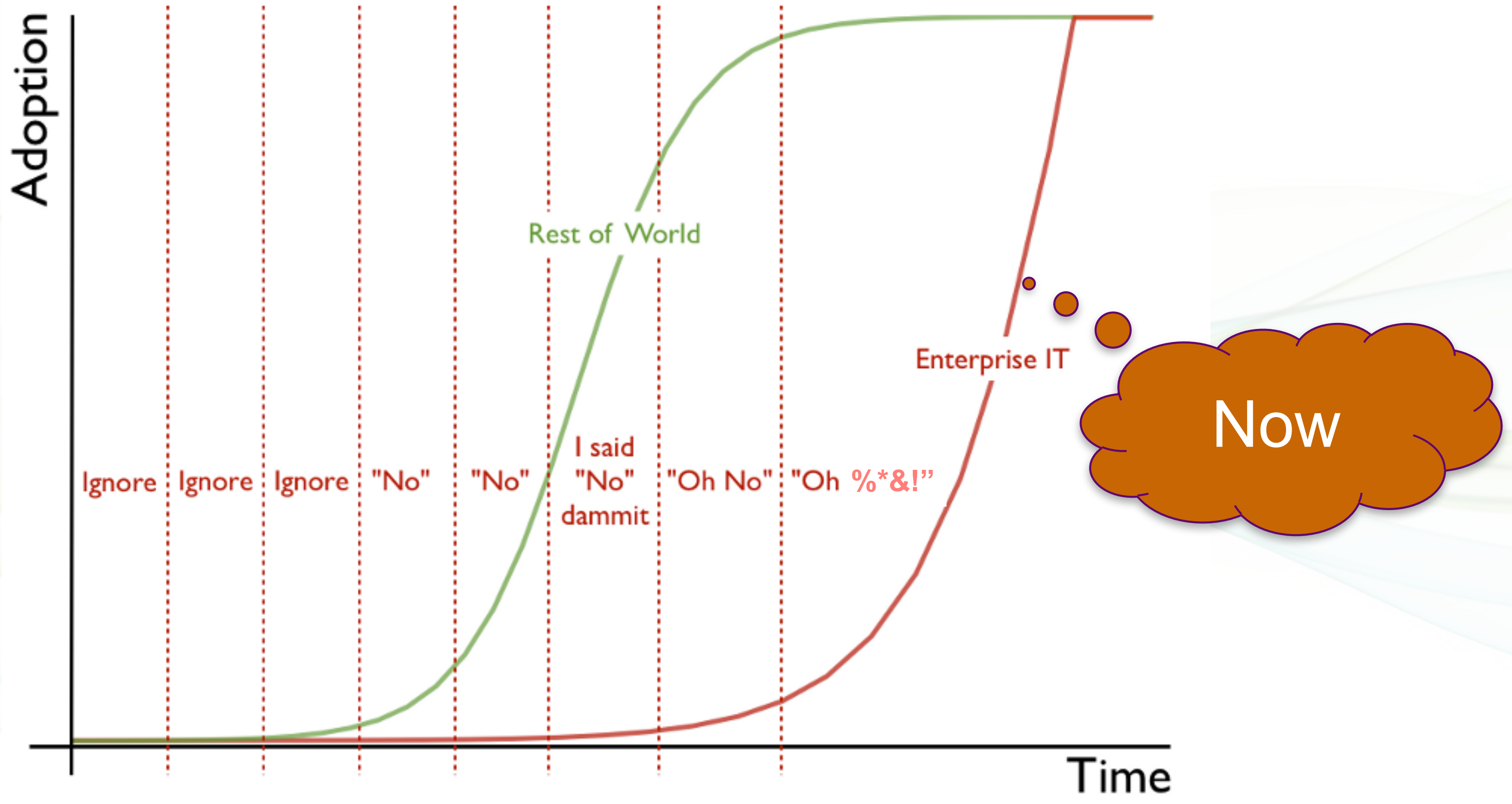
“We’d like to do that but can’t”
– 2012

“We’re on our way using Netflix OSS code”
– 2013

## What I learned from my time at Netflix

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process, no hand-offs between teams
- Freedom and responsibility culture
- Don’t do your own undifferentiated heavy lifting
- Use simple patterns automated by tooling
- Self service cloud makes impossible things instant

# Enterprise IT Adoption of Cloud



By Simon Wardley http://enterpriseitadoption.com/

## Speed

## Innovation

# New ideas

## New products

## What separates incumbents from disruptors?

## Assumptions

## Optimizations

*“It isn't what we don't know that gives us trouble, it's what we know that ain't so.”*

Will Rogers

http://www.brainyquote.com/quotes/quotes/w/willrogers385286.html

## Incumbents follow the $$$

Market size lags disruption because high price products are replaced by low priced products

## Disruptors find what used to be expensive

Learn to waste them to save money elsewhere

## Examples

## Solid State Disk

## Storage systems assume random reads are expensive

Decades of filesystems and storage array development based on spinning rust

RR is free

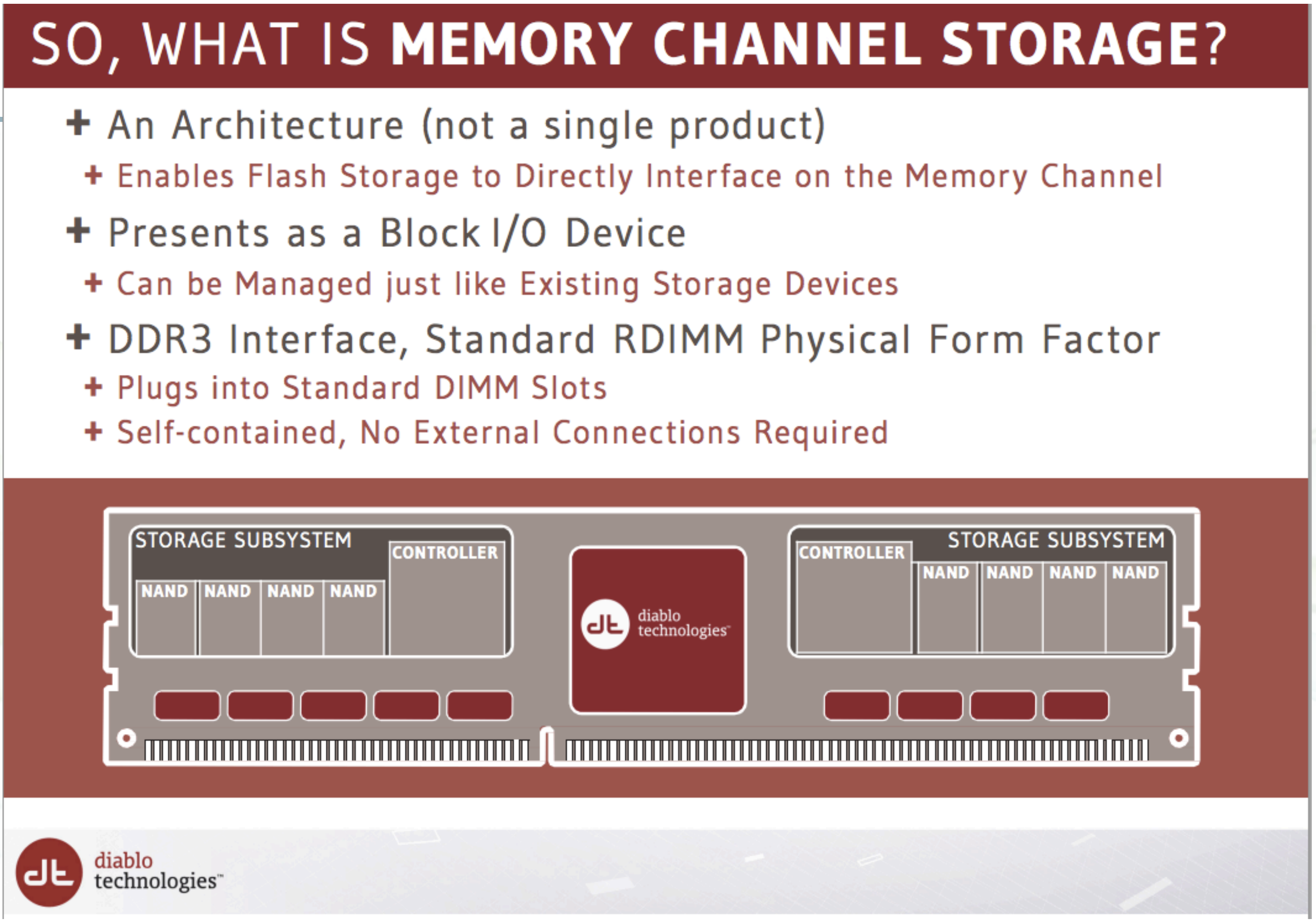
Immutable writes

Log-merge

SSD works best for random reads and sequential writes. Bad for updates.

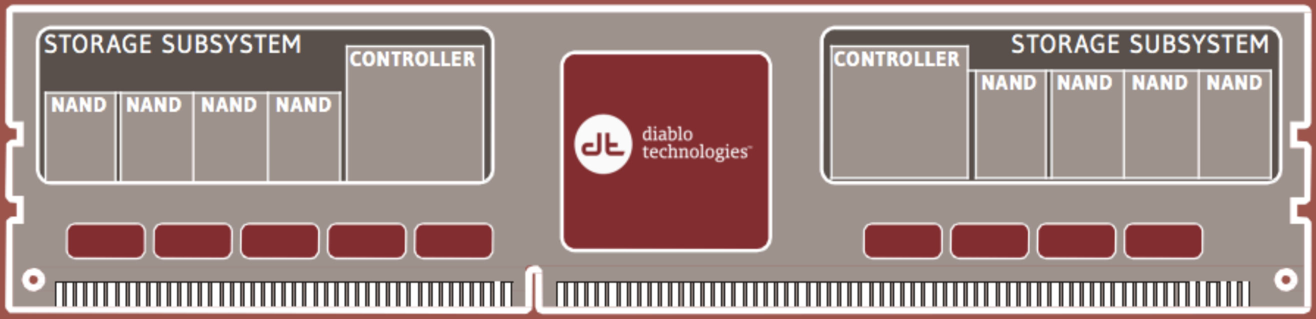
## SSD packaging as disk, as PCI card now as memory DIMM

Each generation reduces overhead and improves price/performance



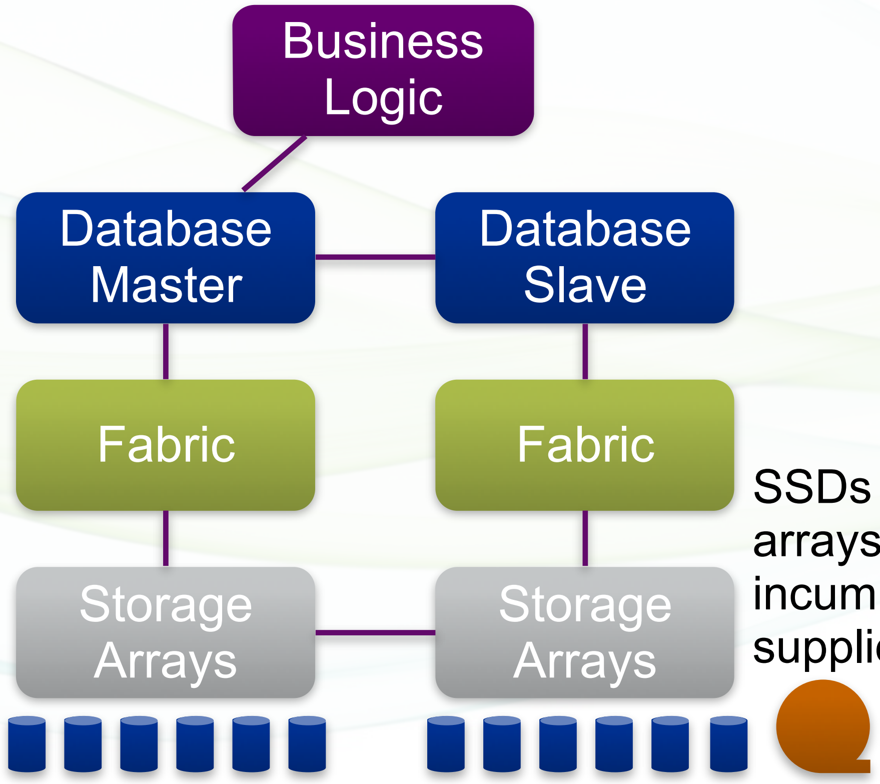
## SO, WHAT IS **MEMORY CHANNEL STORAGE?**

- An Architecture (not a single product)
  - Enables Flash Storage to Directly Interface on the Memory Channel
- Presents as a Block I/O Device
  - Can be Managed just like Existing Storage Devices
- DDR3 Interface, Standard RDIMM Physical Form Factor
  - Plugs into Standard DIMM Slots
  - Self-contained, No External Connections Required

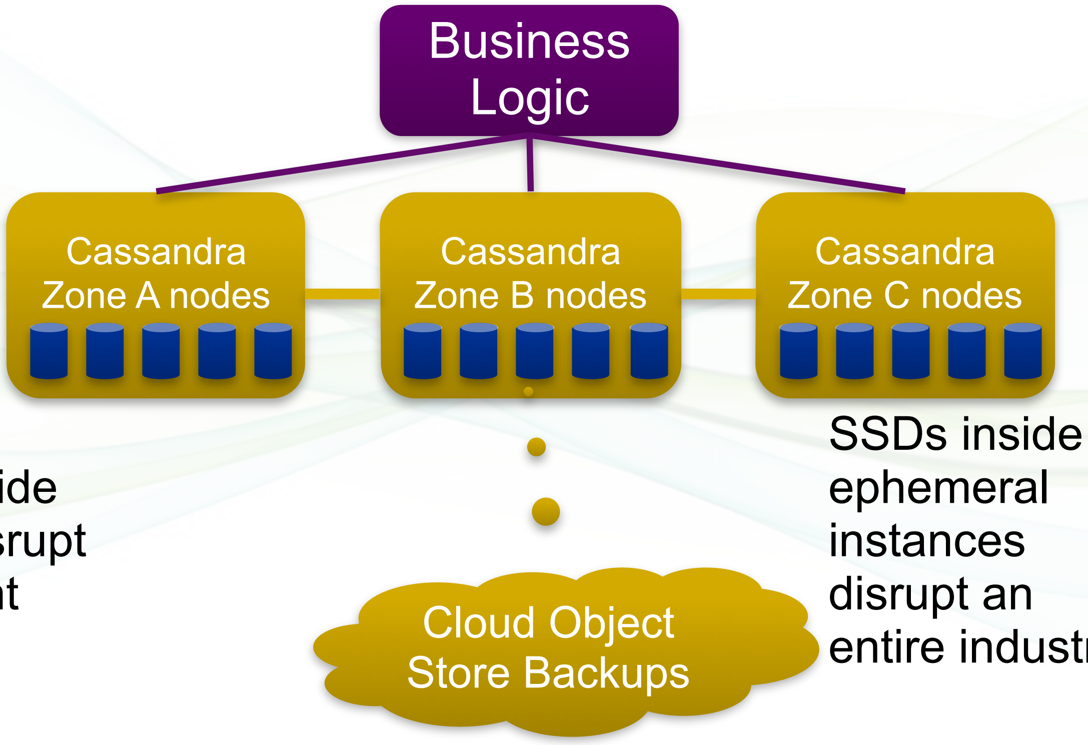


diablo technologies™

## Traditional vs. Cloud Native Storage Architectures



SSDs inside arrays disrupt incumbent suppliers



SSDs inside ephemeral instances disrupt an entire industry

## How to Scale Storage Beyond Ludicrous

- Cassandra scalability
  - Linear scale up benchmarked and seen in production
  - Hundreds of nodes per cluster in common use today
  - Thousands of nodes per cluster actively being tested and used
- Cassandra scale using high end AWS storage instances
  - EC2 i2.8xlarge - over 300,000 iops read or write, 6.4TB of SSD
  - 100 nodes = 30 million iops and 640 TB - Ludicrous
  - 1000 nodes = 300 million iops and 6.4 PB - Plaid!

http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html

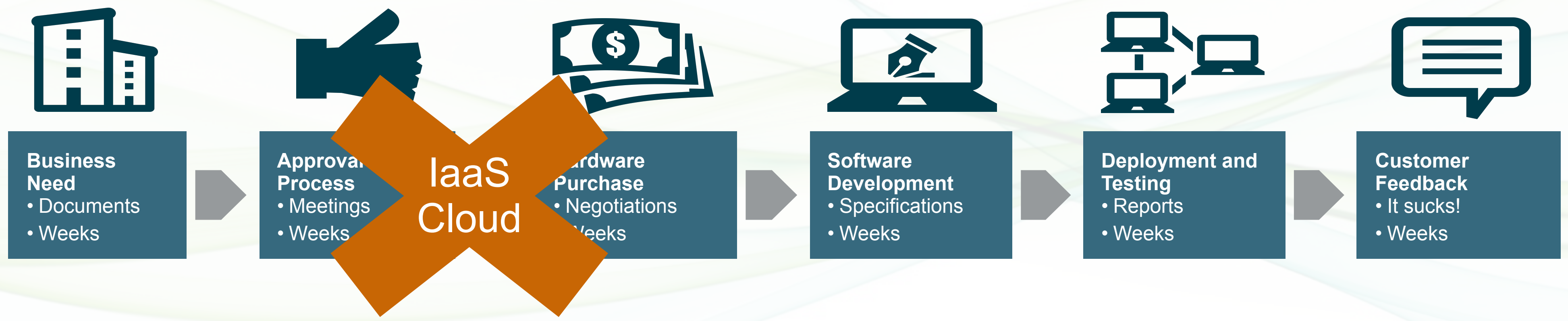
## Disruptor Cassandra

Perfect match for SSD, no write amplification, no updates, scales to plaid

## Product Development

## Assumption: Process prevents problems

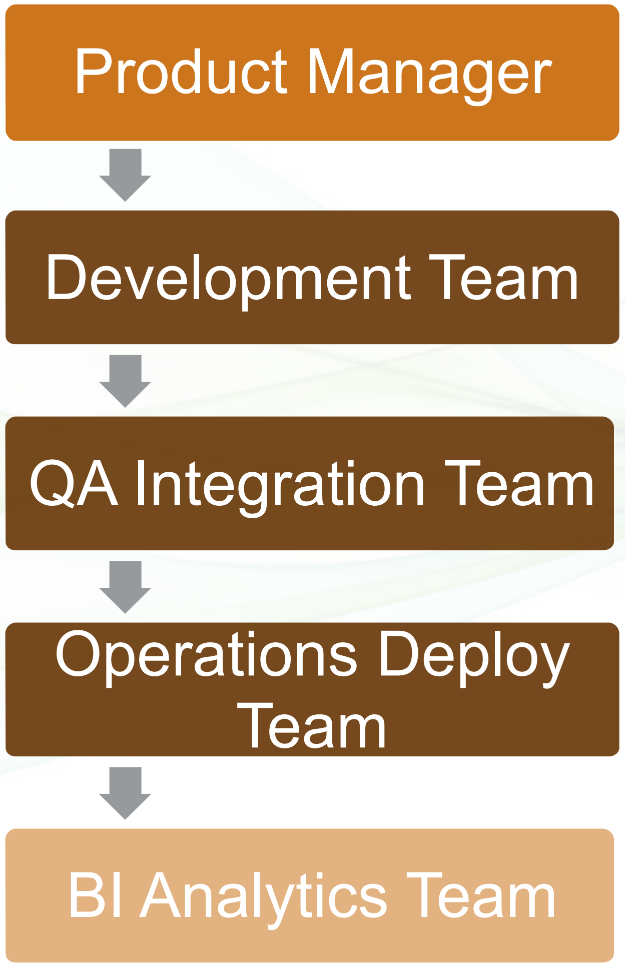
## Non-Cloud Product Development



Hardware provisioning is undifferentiated heavy lifting – replace it with IaaS

Months before you find out whether the product meets the need

## Process Hand-Off Steps for Product Development on IaaS



## IaaS Based Product Development

etc…

**Business Need**
- Documents
- Weeks

**Software Development**
- Specifications
- Weeks

**De… …g**
- Re…
- Day…

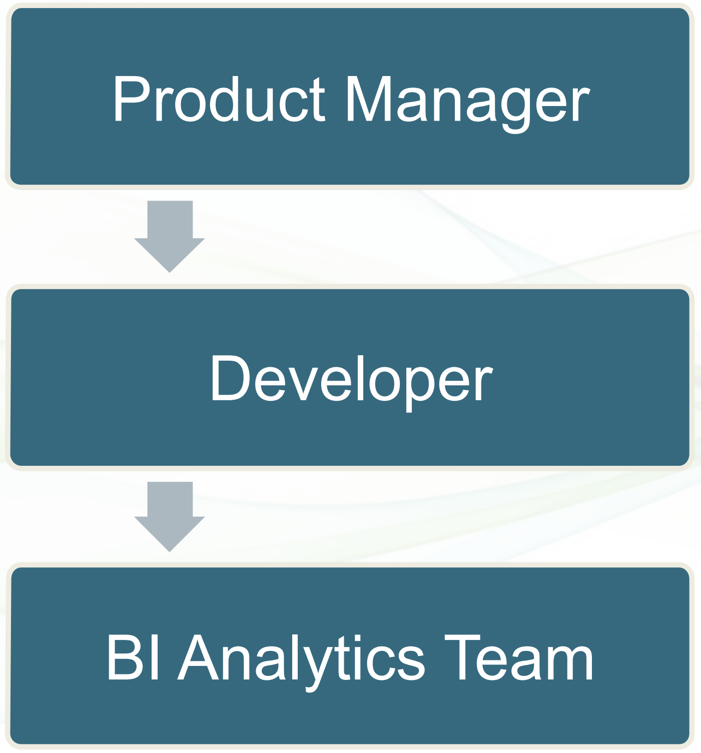
PaaS Cloud

**Customer Feedback**
- It sucks!
- Days

Software provisioning is undifferentiated heavy lifting – replace it with PaaS

Weeks before you find out whether the product meets the need

## Process Hand-Off Steps for Feature Development on PaaS



# PaaS Based Product Feature Development

heroku · CLOUD FOUNDRY · Google App Engine · NETFLIX OSS · apcera · docker · CONTINUUITY · etc…

**Business Need**
- Discussions
- Days

**S t**
- Co
- D

SaaS/
BPaaS
Cloud

**Customer Feedback**
- Fix this Bit!
- Hours

Building your own business apps is undifferentiated heavy lifting – use SaaS

BV

Days before you find out whether the feature meets the need

## SaaS Based Business App Development

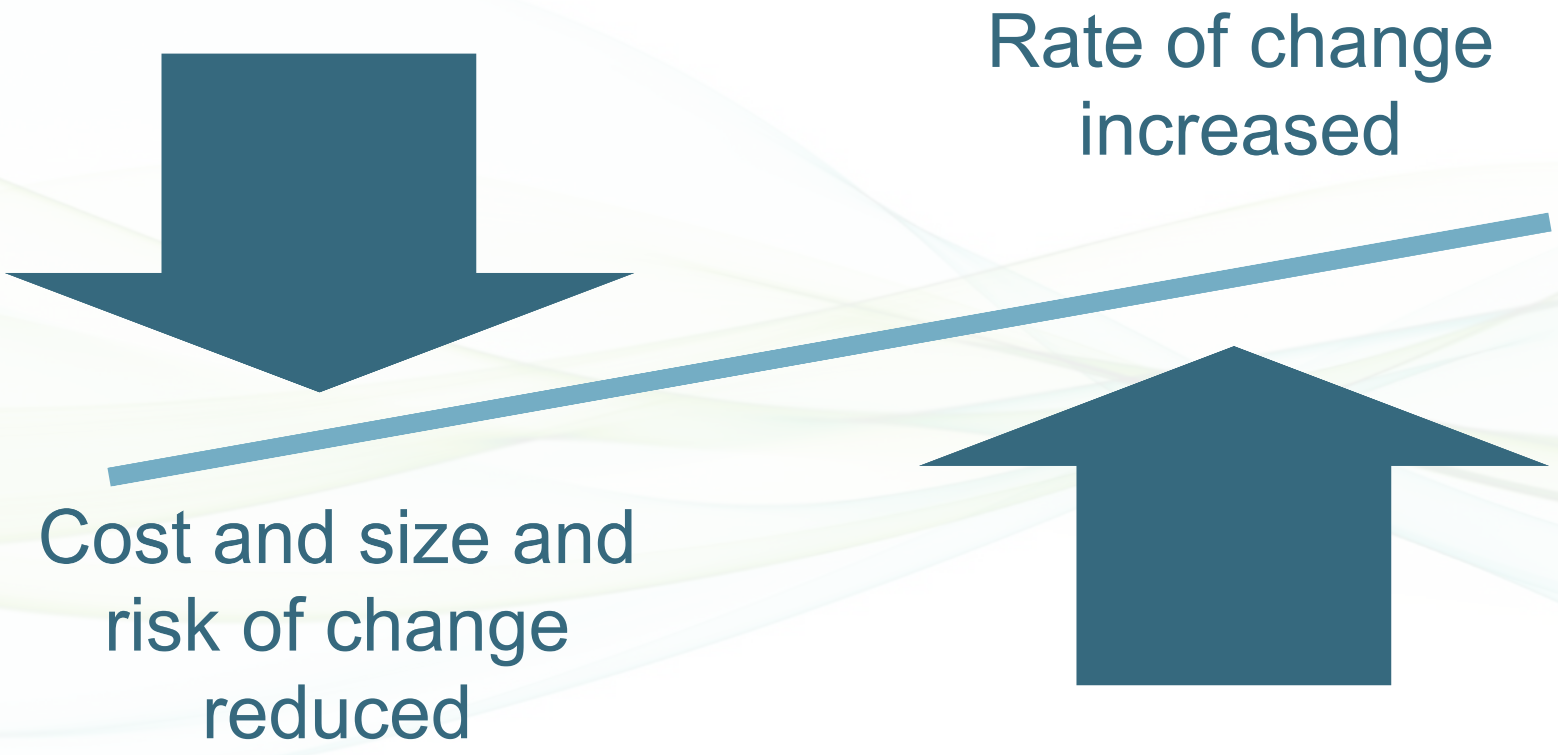
salesforce  mx mendix the app platform  platfora  and thousands more…

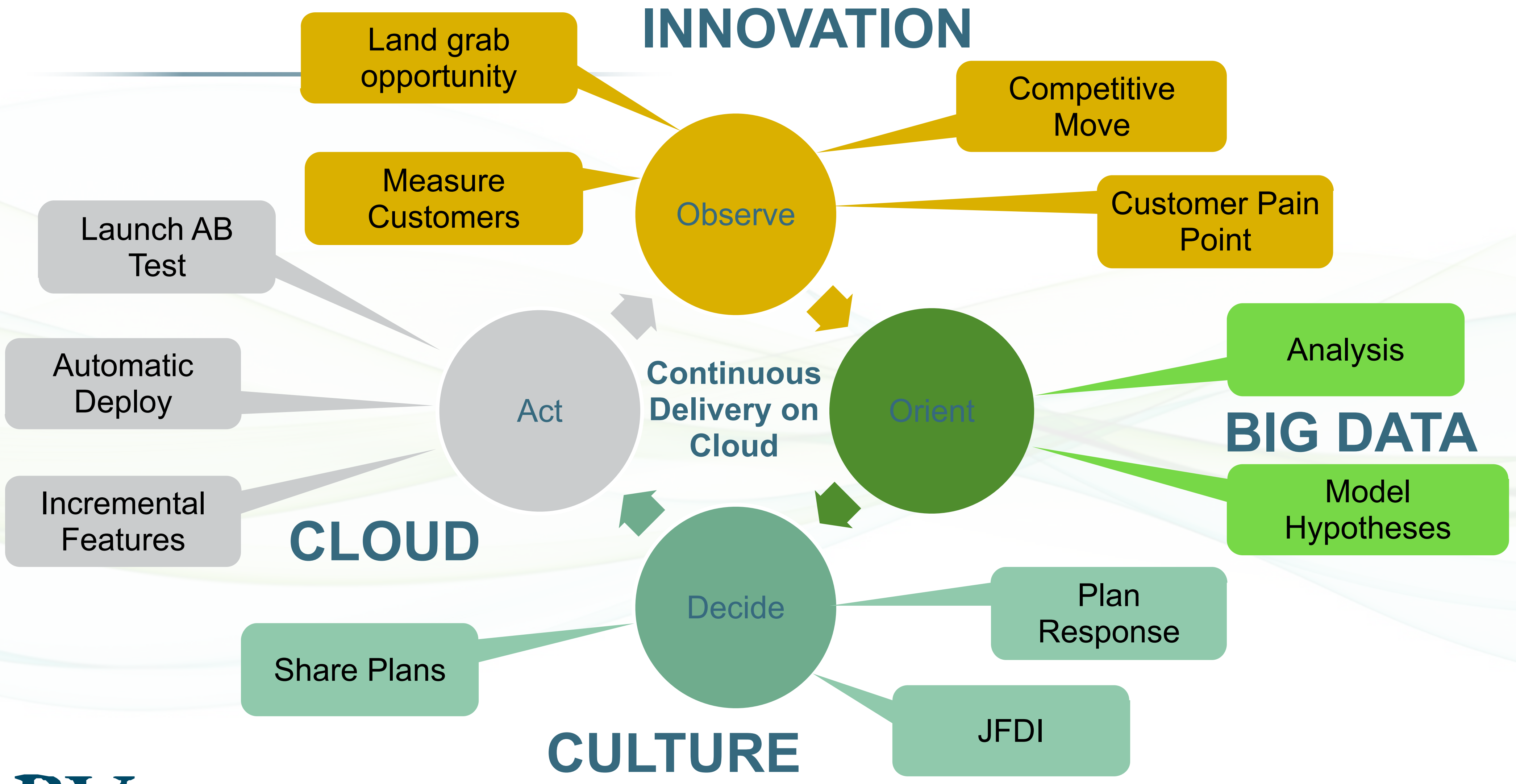
**Business Need**
•GUI Builder
•Hours

**Customer Feedback**
•Fix this bit!
•Seconds

Hours before you find out whether the feature meets the need

## What Happened?




INNOVATION
Land grab opportunity
Measure Customers
Competitive Move
Customer Pain Point
Observe
Launch AB Test
Automatic Deploy
Incremental Features
Act
Continuous Delivery on Cloud
Orient
Analysis
BIG DATA
Model Hypotheses
CLOUD
Decide
Plan Response
Share Plans
JFDI
CULTURE

## Note: Non-Destructive Production Updates

- “Immutable Code” Service Pattern
  - Existing services are unchanged, old code remains in service
  - New code deploys as a new service group
  - No impact to production until traffic routing changes
- A|B Tests, Feature Flags and Version Routing control traffic
  - First users in the test cell are the developer and test engineers
  - A cohort of users is added looking for measurable improvement
  - Finally make default for everyone, keeping old code for a while

# Disruptor
# Continuous Delivery

## Development and Operations

Another disruptive example, if you assume they don’t mix…

## Developers make code

## Operations run code

## It can take weeks to get a VM after a developer files a ticket…

## But if operations is a self service API…

## Developers run their own code

## Developers are on call

## Developers have freedom

## Developers have incentives to be responsible

Avoids the externalities of over-dependence on operations to fix everything

## Less down time

## No meetings

## DevOps is a re-org, not a new team to hire

For most companies, the cultural transformation needed to do DevOps is the blocker

# Disruptor
# High Trust Culture
# DevOps

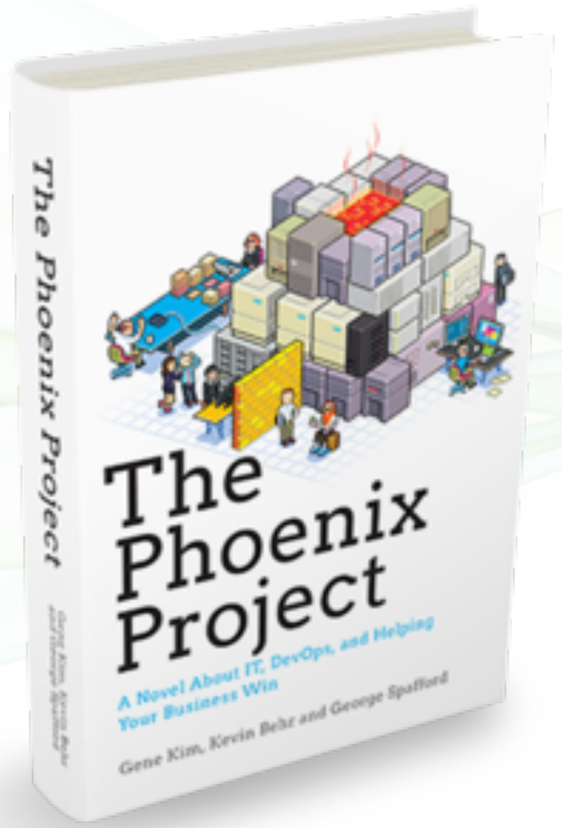
Give up central coordination and control, to get speed and align incentives

## It’s what you know that isn’t so…

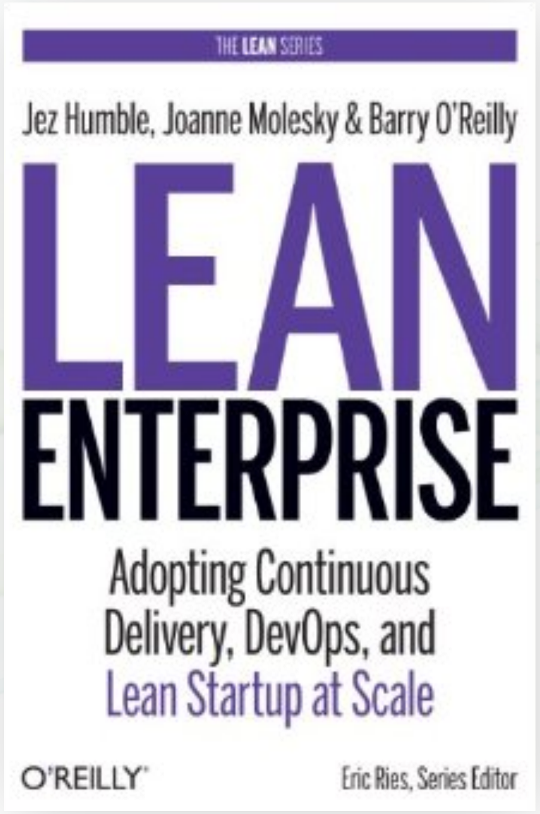
- Make your assumptions explicit
- Extrapolate trends to the limit
- Listen to non-customers
- Follow developer adoption, not IT spend
- Map evolution of products to services to utilities
- Re-organize your teams for speed of execution

## How do we get there?

## *"This is the IT swamp draining manual for anyone who is neck deep in alligators."*


The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr and George Spafford

## Once you’re out of the swamp, read this…


THE LEAN SERIES

Jez Humble, Joanne Molesky & Barry O'Reilly

LEAN ENTERPRISE

Adopting Continuous Delivery, DevOps, and Lean Startup at Scale

O'REILLY

Eric Ries, Series Editor

## Open Source Ecosystems

- The most advanced, scalable and stable code you can get is OSS
- No procurement cycle, fix and extend it yourself
- Github is a developer’s online resume
- Github is also your company’s online resume!
- Extensible platforms create ecosystems
- Give up control to get ubiquity – Apache license

Innovate, Leverage and Commoditize

## Cloud Native for High Availability

- Business logic isolation in stateless micro-services
- Immutable code with instant rollback
- Auto-scaled capacity and deployment updates
- Distributed across availability zones and regions
- De-normalized single function NoSQL data stores
- See over 40 NetflixOSS projects at netflix.github.com
- Get “Technical Indigestion” trying to keep up with techblog.netflix.com

## A Microservice Definition

Loosely coupled service oriented architecture with bounded contexts

See http://en.wikipedia.org/wiki/Domain-driven_design for discussion of bounded contexts

## Scaling Continuous Delivery Models

***Monolithic***

- Devs book a train ticket
- Everyone runs the monolith
- Queue for the next train
- Coordination chat session
- Need to learn deploy process
- Copy code to existing servers
- Few concurrent versions
- Tens of monolithic updates/day maximum
- Roll-forward only
- “Done” is released to prod

***Microservices***

- Everyone has their own build
- Dev runs their own microservice
- No waiting, no meetings
- API call to update prod timeline
- Automated hands-off deploy
- Immutable code on new servers
- Unlimited concurrent versions
- 100s of independent updates
- Roll-back in seconds
- “Done” is retired from prod

## Separate Concerns Using Micro-services

- Invert Conway's Law – teams own service groups and backend stores
- One "verb" per single function micro-service, size doesn't matter
- One developer independently produces a micro-service
- Each micro-service is it's own build, avoids trunk conflicts
- Deploy in a container: Tomcat, AMI or Docker, whatever…
- Stateless business logic. Cattle, not pets.
- Stateful cached data access layer can use ephemeral instances

http://en.wikipedia.org/wiki/Conway's_law

## Microservices Development Architecture

- Client libraries
  - Even if you start with a raw protocol, a client side driver is the end-state
  - Best strategy is to own your own client libraries from the start
- Multithreading and Non-blocking Calls
  - Reactive model RxJava uses Observable to hide concurrency cleanly
  - Netty can be used to get non-blocking I/O speedup over Tomcat container
- Circuit Breakers – See Fluxcapacitor.com for code
  - NetflixOSS Hystrix, Turbine, Latency Monkey, Ribbon/Karyon
  - Also look at Finagle/Zipkin from Twitter

## Microservice Datastores

- Book: Refactoring Databases
  - SchemaSpy to examine schema structure
  - Denormalization into one datasource per table or materialized view
- Polyglot Persistence
  - Use a mixture of database technologies, behind REST data access layers
  - See NetflixOSS Storage Tier as a Service HTTP (staash.com) for MySQL and C*
- CAP – Consistent or Available when Partitioned
  - Look at Jepsen torture tests for common systems aphyr.com/tags/jepsen
  - There is no such thing as a consistent distributed system, get over it…

## Strategies for impatient product managers

- Carrot

  *“This new feature you want will be ready faster as a microservice”*

- Stick

  *“This new feature you want will only be implemented in the new microservice based system”*

- Shiny Object

  *“Why don’t you concentrate on some other part of the system while we get the transition done?”*

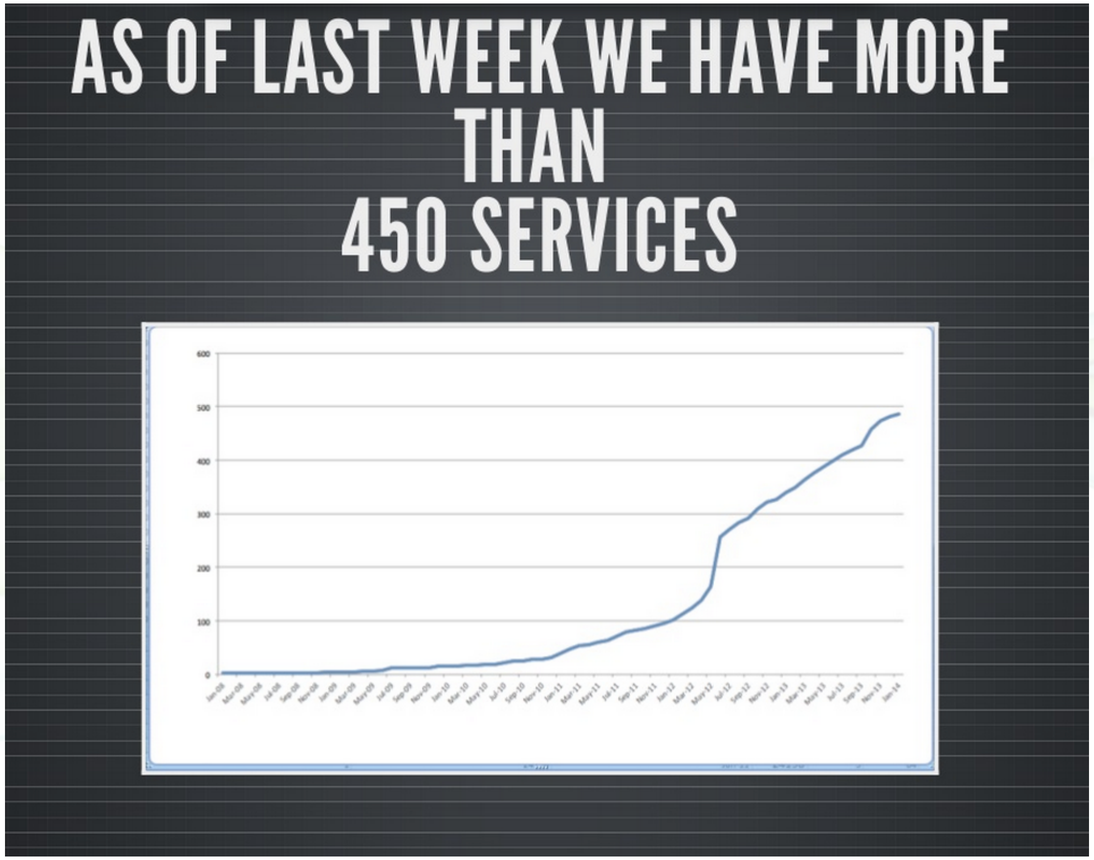
## Monitoring and Microservices

## Issues with Continuous Delivery and Microservices

- High rate of change
  - Code pushes can cause floods of new instances and metrics
  - Short baseline for alert threshold analysis – everything looks unusual
- Ephemeral Configurations
  - Short lifetimes make it hard to aggregate historical views
  - Hand tweaked monitoring tools take too much work to keep running
- Microservices with complex calling patterns
  - End-to-end request flow measurements are very important
  - Request flow visualizations get overwhelmed

## Microservice Based Architectures

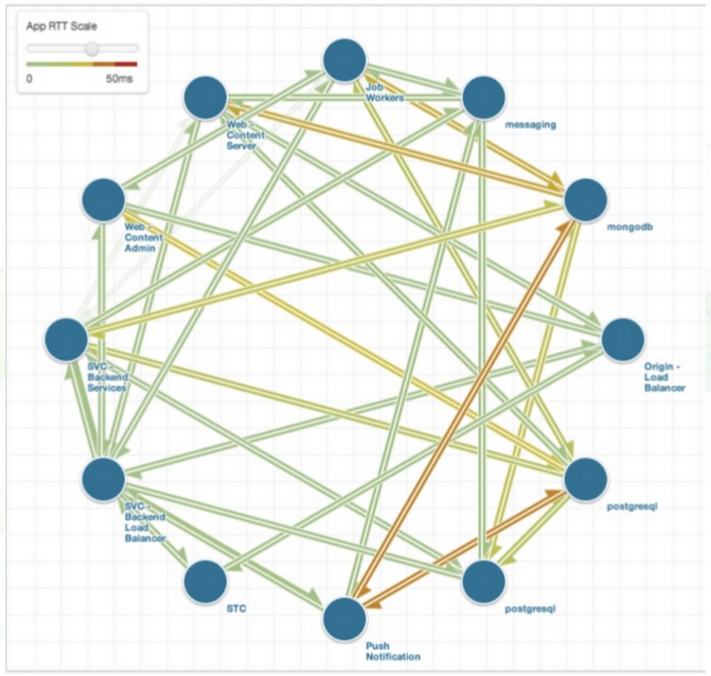
***From a Gilt Groupe Presentation***



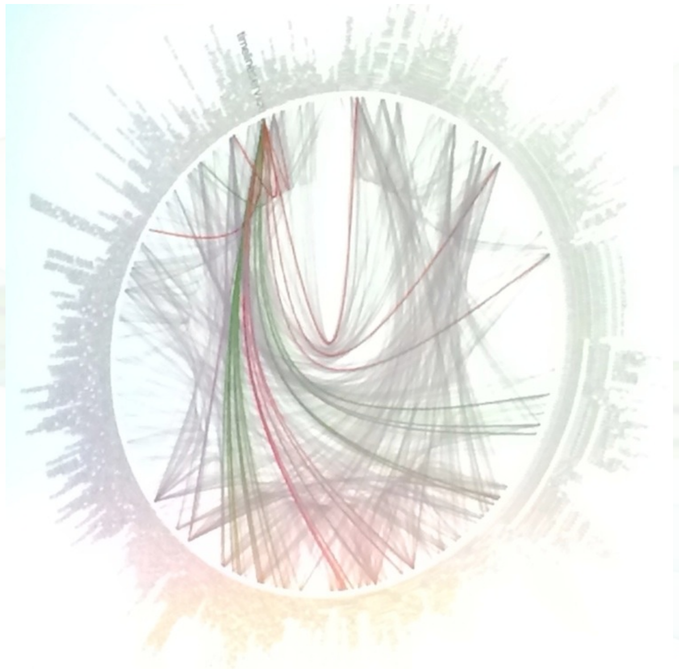
See http://www.slideshare.net/LappleApple/gilt-from-monolith-ruby-app-to-micro-service-scala-service-architecture

## “Death Star” Architecture Diagrams

Netflix



Gilt Groupe (12 of 450)



Twitter

As visualized by Appdynamics, Boundary.com and Twitter internal tools

## Monitoring Micro-services

*Visualizing the request flow*

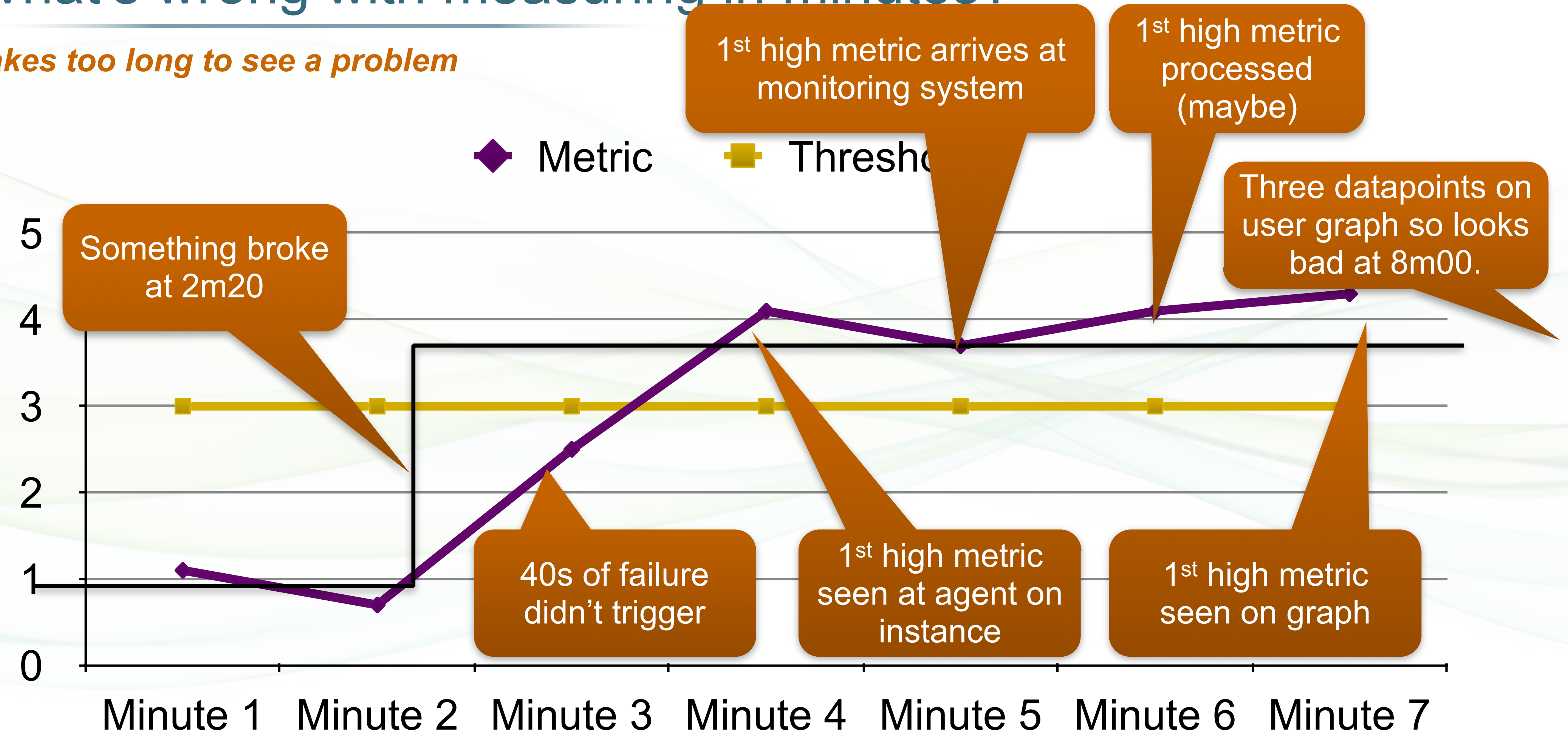
- Appdynamics
  - Instrument the JVM to capture everything including traffic flows
  - Insert tag for every http request with a header annotation guid
  - Visualize the over-all flow or the business transaction flow
- Boundary.com and Lyatiss CloudWeaver
  - Instrument the packet flows across the network
  - Capture the zone and region config from cloud APIs and tags
  - Correlate, aggregate and visualize the traffic flows
- Instrumented PaaS Communication Mechanisms
  - CloudFoundry and Apcera route all traffic through NATS
  - NetflixOSS ribbon client and karyon server http annotation guid
  - In-band mechanisms can scale beyond capabilities of centralized tools

## Continuous Delivery and DevOps Implications

- Changes are smaller but more frequent
- Individual changes are more likely to be broken
- Changes are normally deployed by developers
- Feature flags are used to enable new code
- Instant detection and rollback matters much more

## What's wrong with measuring in minutes?

***Takes too long to see a problem***

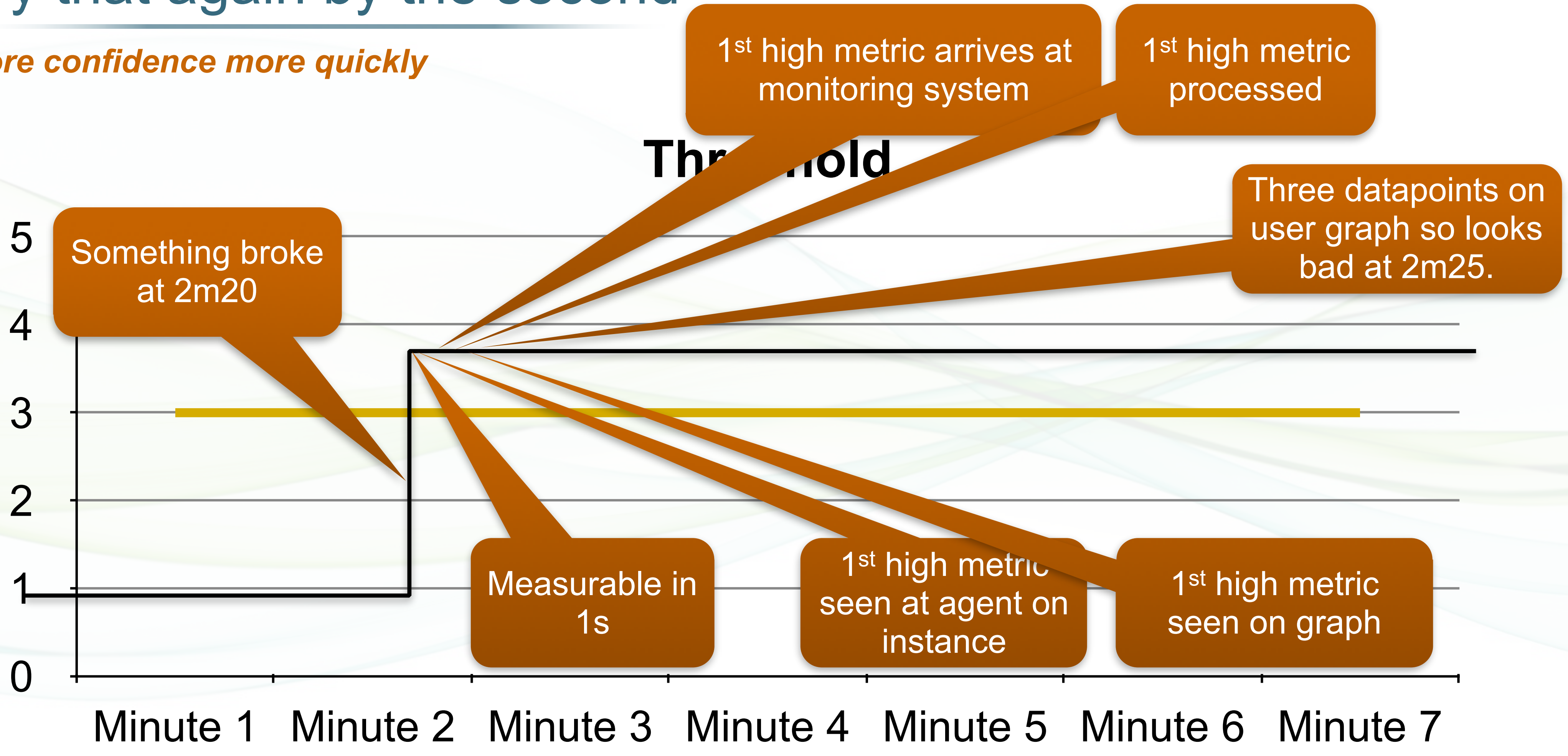


## Whoops! I didn’t mean that! Reverting…

Not cool if it takes 5 minutes to see it failed and 5 more to see a fix
No-one notices if it only takes 5 seconds to detect and 5 to see a fix

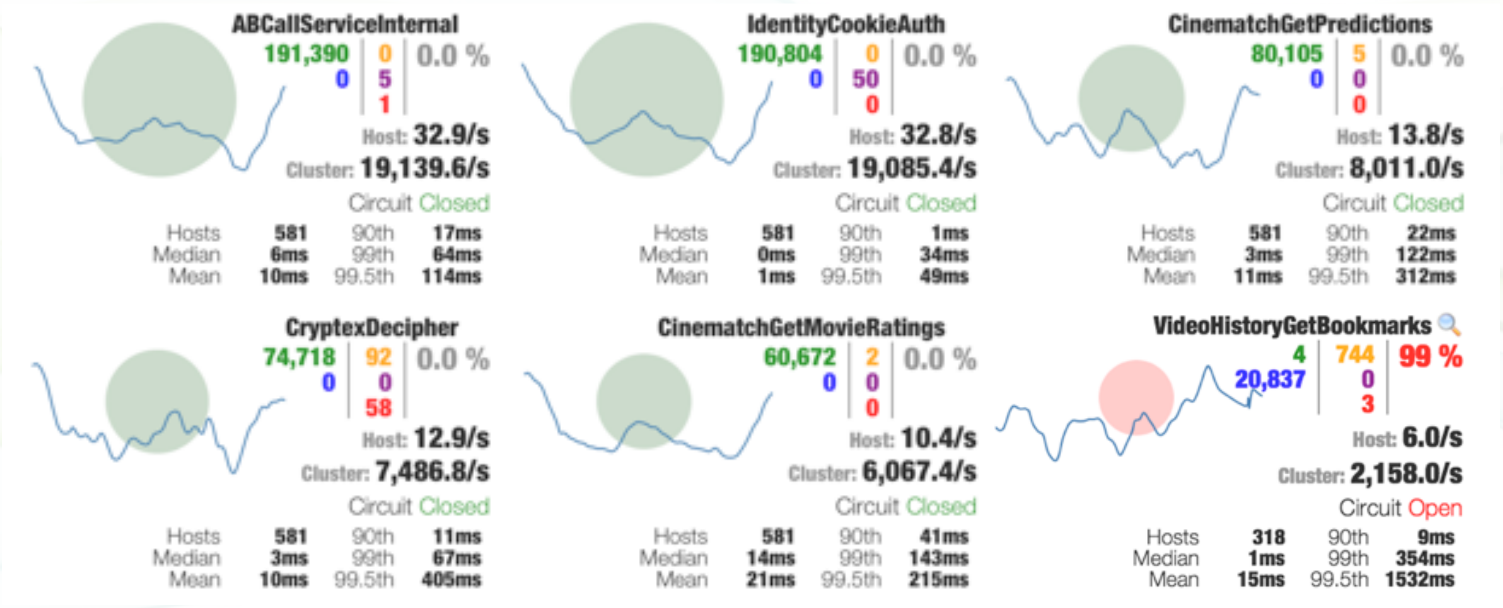
## Try that again by the second

***More confidence more quickly***



# NetflixOSS Hystrix / Turbine Circuit Breaker Monitoring

***Streaming metrics directly from services to a web browser each second***



**ABCallServiceInternal**
191,390 0 | 0 5 1 | 0.0 %
Host: 32.9/s
Cluster: 19,139.6/s
Circuit Closed
Hosts 581 90th 17ms
Median 6ms 99th 64ms
Mean 10ms 99.5th 114ms

**IdentityCookieAuth**
190,804 0 | 0 50 0 | 0.0 %
Host: 32.8/s
Cluster: 19,085.4/s
Circuit Closed
Hosts 581 90th 1ms
Median 0ms 99th 34ms
Mean 1ms 99.5th 49ms

**CinematchGetPredictions**
80,105 0 | 5 0 0 | 0.0 %
Host: 13.8/s
Cluster: 8,011.0/s
Circuit Closed
Hosts 581 90th 22ms
Median 3ms 99th 122ms
Mean 11ms 99.5th 312ms

**CryptexDecipher**
74,718 0 | 92 0 58 | 0.0 %
Host: 12.9/s
Cluster: 7,486.8/s
Circuit Closed
Hosts 581 90th 11ms
Median 3ms 99th 67ms
Mean 10ms 99.5th 405ms

**CinematchGetMovieRatings**
60,672 0 | 2 0 0 | 0.0 %
Host: 10.4/s
Cluster: 6,067.4/s
Circuit Closed
Hosts 581 90th 41ms
Median 14ms 99th 143ms
Mean 21ms 99.5th 215ms

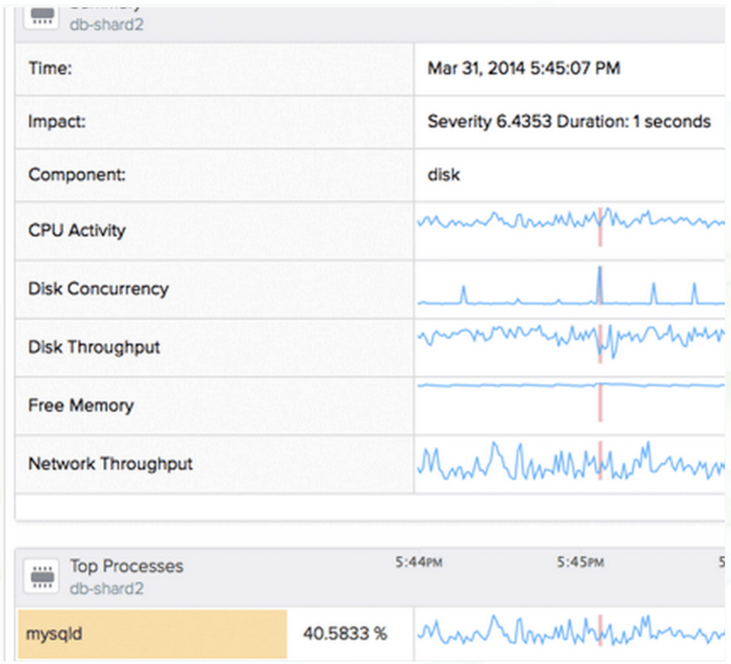
**VideoHistoryGetBookmarks**
4 20,837 | 744 0 3 | 99 %
Host: 6.0/s
Cluster: 2,158.0/s
Circuit Open
Hosts 318 90th 9ms
Median 1ms 99th 354ms
Mean 15ms 99.5th 1532ms

http://techblog.netflix.com/2012/12/hystrix-dashboard-and-turbine.html

# Latest SaaS Based Monitoring Products

## *Seeing Problems In Seconds*

VividCortex



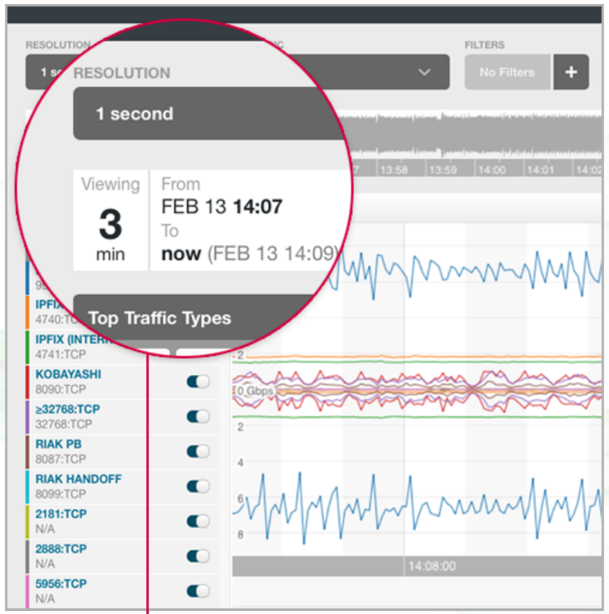
db-shard2

| | |
|---|---|
| Time: | Mar 31, 2014 5:45:07 PM |
| Impact: | Severity 6.4353 Duration: 1 seconds |
| Component: | disk |
| CPU Activity | |
| Disk Concurrency | |
| Disk Throughput | |
| Free Memory | |
| Network Throughput | |

Top Processes
db-shard2

| | | 5:44PM 5:45PM |
|---|---|---|
| mysqld | 40.5833 % | |

boundary



1-second data collection and real-time streaming processing on all components of the application stack

BV

www.vividcortex.com and www.boundary.com

## Metric to display latency needs to be less than human attention span (~10s)

## Summary

- Speed wins in the marketplace
- Remove friction from product development
- High trust, low process
- Freedom and responsibility culture
- Don’t do your own undifferentiated heavy lifting
- Simple patterns automated by tooling
- Microservices for speed and availability

## Separation of Concerns

## Bounded Contexts

## Any Questions?

- Battery Ventures http://www.battery.com
- Adrian's Blog http://perfcap.blogspot.com
- Slideshare http://slideshare.com/adriancockcroft

- Migrating to Microservices – Qcon London - March 6th, 2014
- Monitorama Opening Keynote Portland OR - May 7th, 2014
- GOTO Chicago Opening Keynote May 20th, 2014
- DevOps Summit at Cloud Expo New York – June 10th, 2014
- Qcon New York – June 11th, 2014
- GOTO Copenhagen/Aarhus – Denmark – Oct 25th, 2014