

NETFLIX

OSS

# Globally Distributed Cloud Native Applications at Netflix

June 2013

Adrian Cockcroft

@adrianco #netflixcloud @NetflixOSS

<http://www.linkedin.com/in/adriancockcroft>

Cloud Native

Global Architecture

NetflixOSS Components

Cloud Native

# We are Engineers

We solve hard problems

We build amazing and complex things

We fix things when they break



We strive for perfection

Perfect code

Perfect hardware

Perfectly operated



**Utopia**

But perfection takes too long...

So we compromise

Time to market vs. Quality

Utopia remains out of reach

# Where time to market wins big

Making a land-grab

Disrupting competitors (OODA)

Anything delivered as web services

# How Soon?

Code features in days instead of months

Get hardware in minutes instead of weeks

Incident response in seconds instead of hours

# Tipping the Balance

Utopia

Dystopia

Static

Sooner

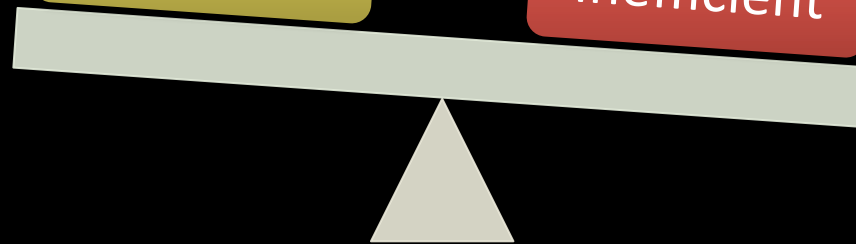
Better

Dynamic

Cheaper

Broken

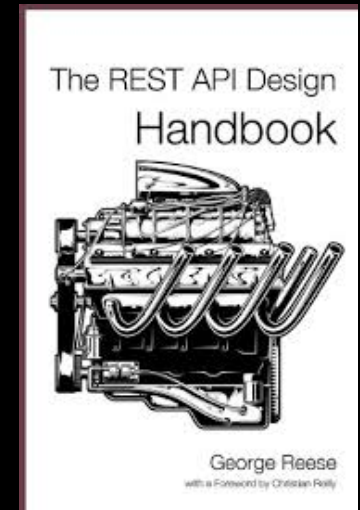
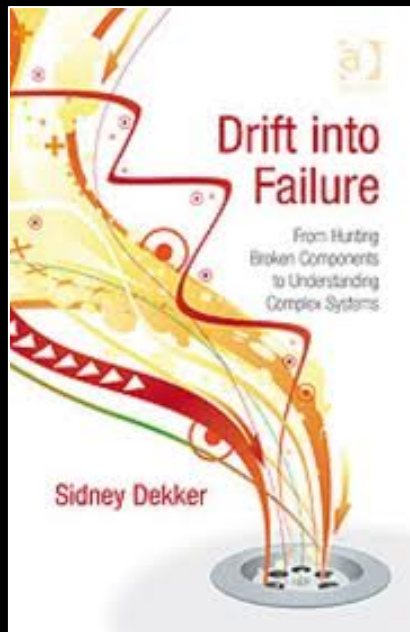
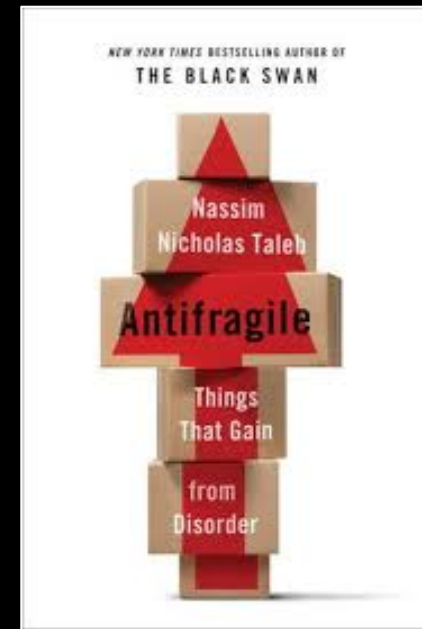
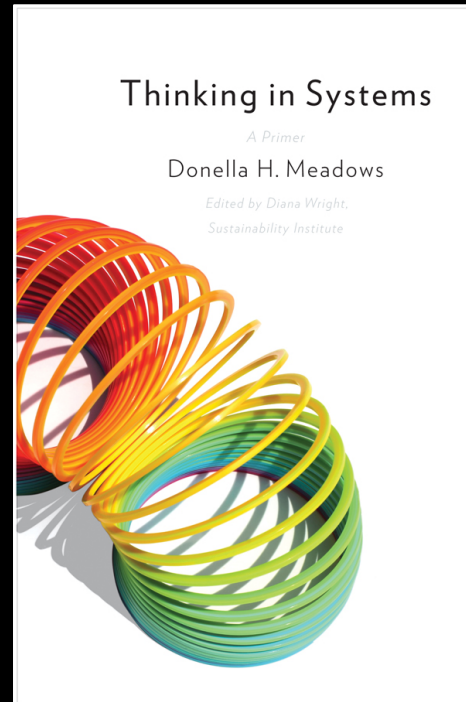
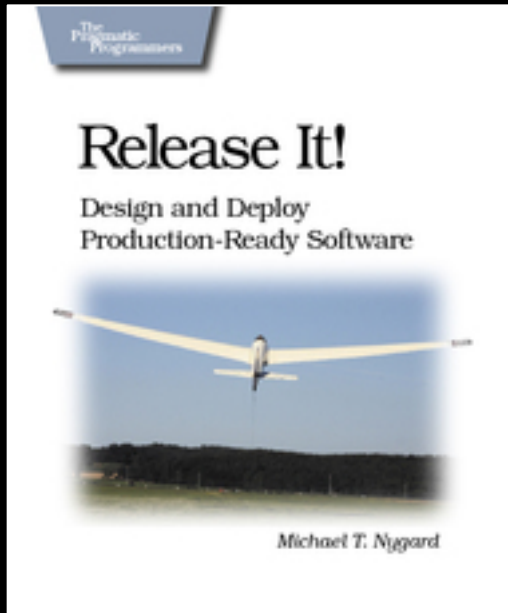
Inefficient



# A new engineering challenge

Construct a highly agile and highly available service from ephemeral and often broken components

# Inspiration



# Netflix Streaming

A Cloud Native Application based on  
an open source platform



# Netflix Member Web Site Home Page

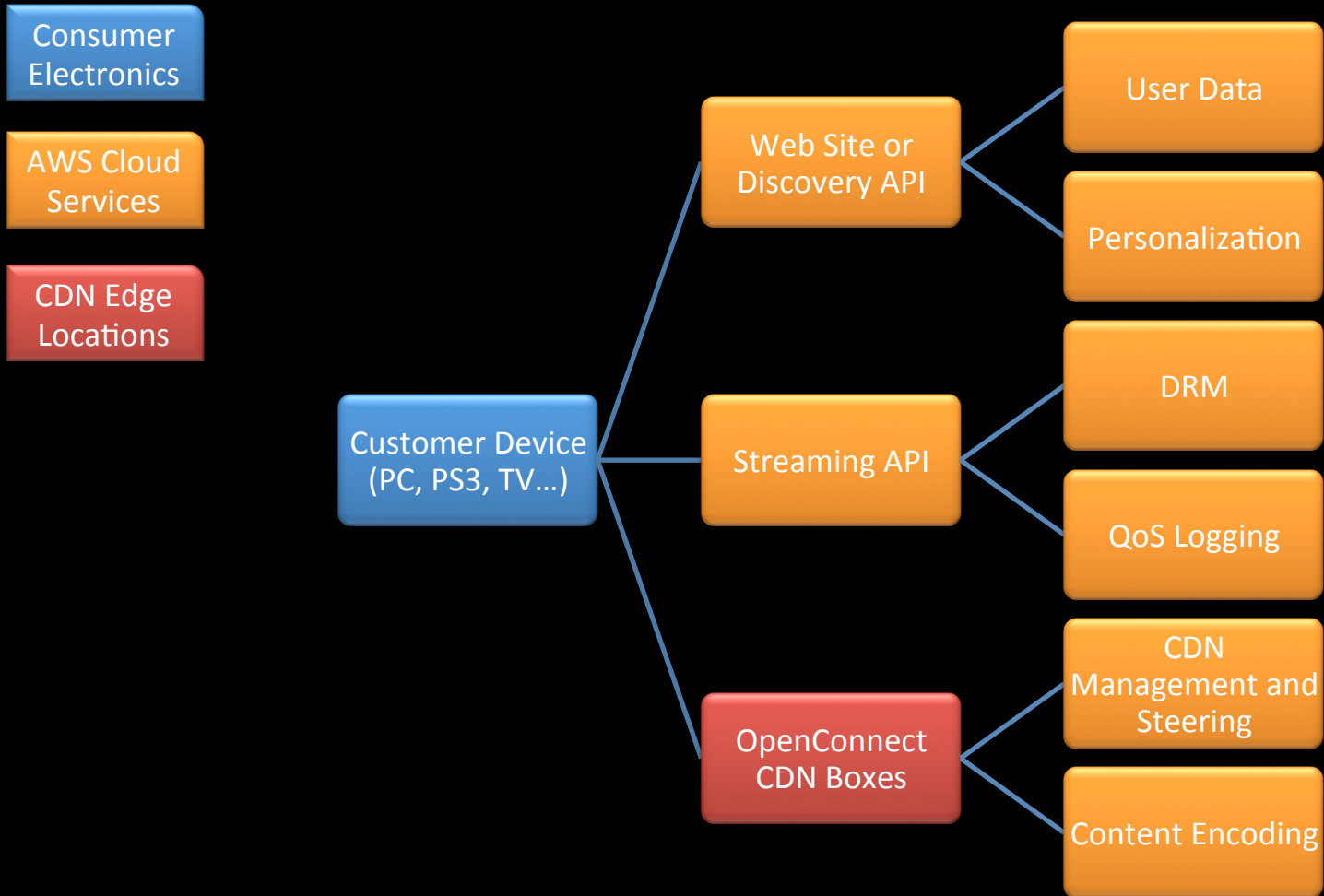
## Personalization Driven – How Does It Work?

The screenshot shows the Netflix member website home page for Adrian Cockcroft. The top navigation bar includes the Netflix logo, the user's name "Adrian Cockcroft", and a link to "Your Account & Help". Below this is a secondary navigation bar with tabs for "Watch Instantly", "Just for Kids", "Browse DVDs", "Your Queue", and "Taste Profile". A search bar on the right contains the text "Movies, TV shows, actors, directors, genres".

The main content area is divided into several sections:

- Recently Watched:** A vertical list of items, including "JOHN MAYALL & THE BLUESBREAKERS AND FRIENDS" and "70th BIRTHDAY CONCERT".
- Top 10 for Adrian:** A horizontal row of ten items: "DEFYING DISEASE TED TALKS", "SAM KINISON BREAKING THE RULES", "ANCIENT INVENTIONS OF WAR, SEX AND CITY LIFE", "ROBOTIC MACHINATIONS TED TALKS", "that Mitchell and Webb look", and "Bartleby".
- Friends' Favorites:** A section titled "Based on these friends:" with two profile pictures. Below are six movie recommendations: "Breaking Bad", "LOST IN TRANSLATION", "THE TERMINATOR", "Audrey BREAKFAST AT TIFFANY'S", "THE HUNT FOR RED OCTOBER", and "GOOD WILL HUNTING".

# How Netflix Streaming Works



Nov  
2012  
Streaming  
Bandwidth  
18x Prime

Rank	Upstream		Downstream		Aggregate		
	Application	Share	Application	Share	Application	Share	
1	BitTorrent	36.8%	Netflix	33.0%	Netflix	28.8%	
2	HTTP	9.83%	YouTube	14.8%	YouTube	13.1%	
3	Skype	4.76%	HTTP	12.0%	HTTP	11.7%	
4	Netflix	4.51%	BitTorrent	5.89%	BitTorrent	10.3%	
5	SSL	3.73%	iTunes	3.92%	iTunes	3.43%	
6	YouTube	2.70%	MPEG	2.22%	SSL	2.23%	
7	PPStream	1.65%	Flash Video	2.21%	MPEG	2.05%	
8	Facebook	1.62%	SSL	1.97%	Flash Video	2.01%	
9	Apple PhotoStream	1.46%	Amazon Video	1.75%	Facebook	1.50%	
10	Dropbox	1.17%	Facebook	1.48%	RTMP	1.41%	
Top 10		68.24%	Top 10		79.01%	Top 10	76.54%

sandvine

Table 3 - Top 10 Peak Period Applications (North America, Fixed Access)

March  
2013  
Mean  
Bandwidth  
+39% 6mo  
25x Prime

Rank	Upstream		Downstream		Aggregate	
	Application	Share	Application	Share	Application	Share
1	BitTorrent	34.81%	Netflix	32.25%	Netflix	28.88%
2	HTTP		YouTube	17.11%	YouTube	15.43%
3	SSL		HTTP	11.11%	HTTP	10.66%
4	Netflix		BitTorrent	5.57%	BitTorrent	9.23%
5	Skype		MPEG	2.58%	SSL	2.39%
6	YouTube		Hulu	2.41%	MPEG	2.30%
7	Facebook		iTunes	1.90%	Hulu	2.16%
8	Apple PI		SSL	1.89%	iTunes	1.71%
9	Dropbox		Flash Video	1.72%	Flash Video	1.53%
10	Carboni		Facebook	1.48%	Facebook	1.52%
Top 10		67.38%			78.03%	75.82%



Free Two-Day  
Shipping

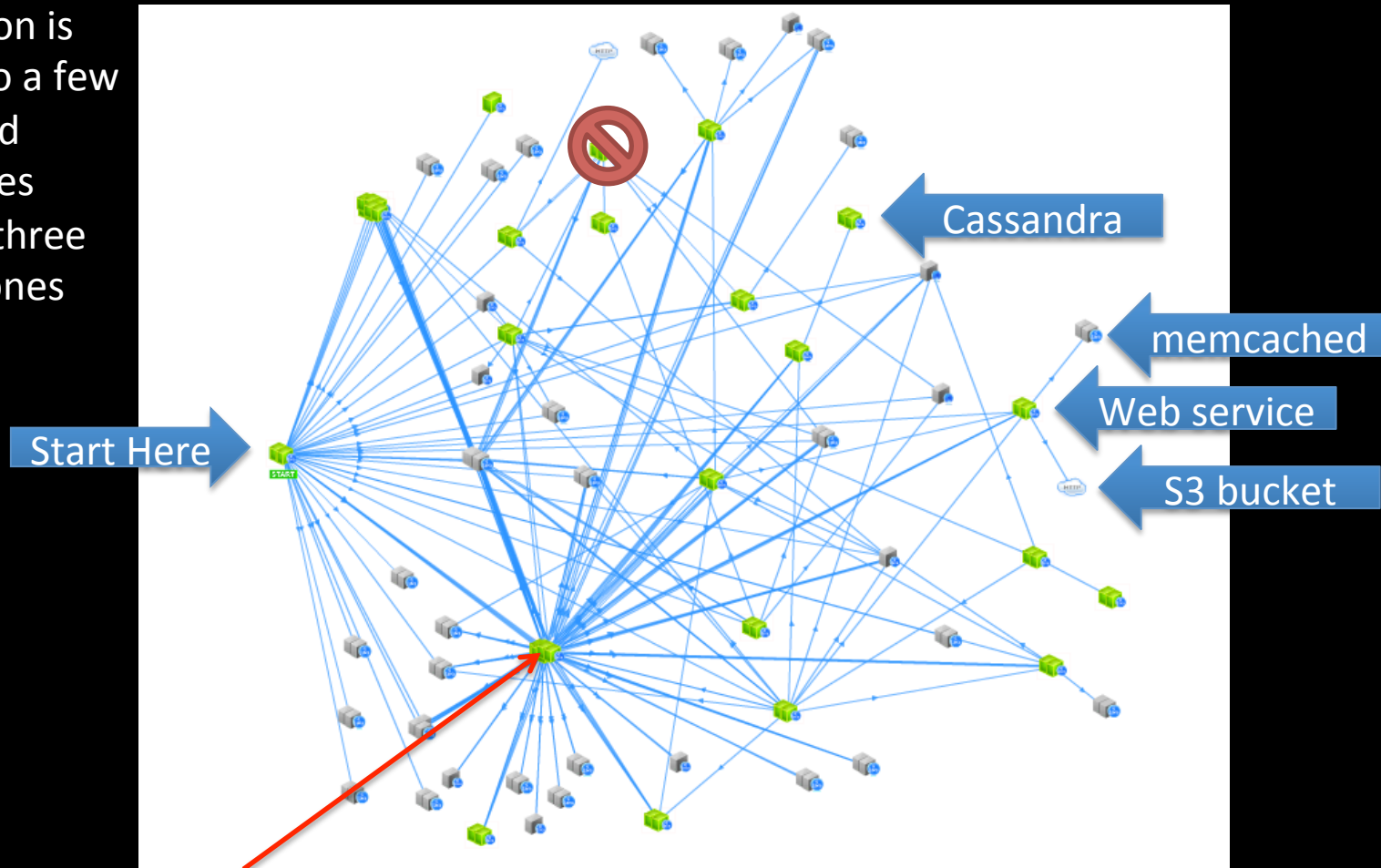
Amazon Video 1.31%

sandvine

# Real Web Server Dependencies Flow

(Netflix Home page business transaction as seen by AppDynamics)

Each icon is three to a few hundred instances across three AWS zones

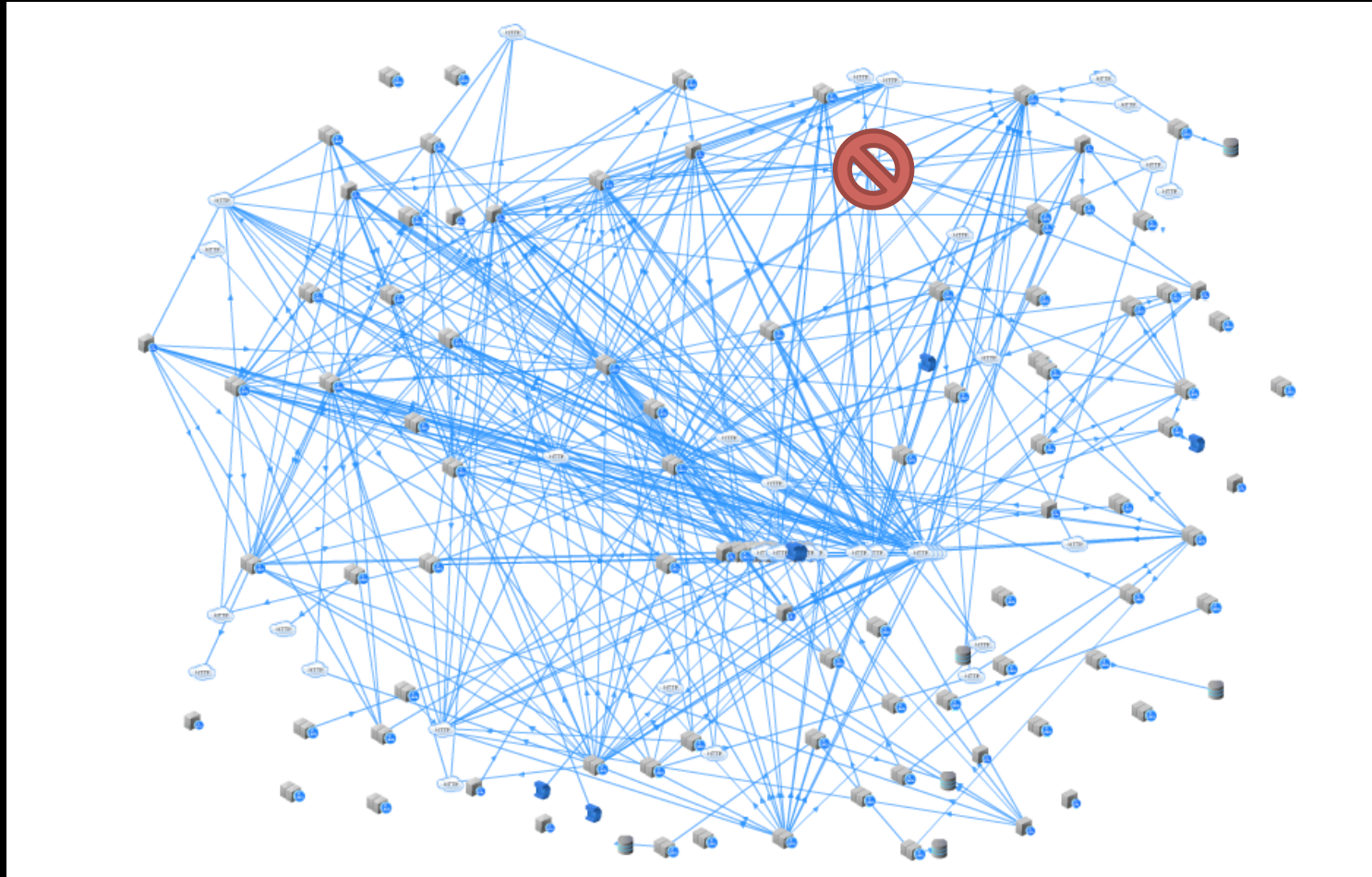


Personalization movie group choosers  
(for US, Canada and Latam)



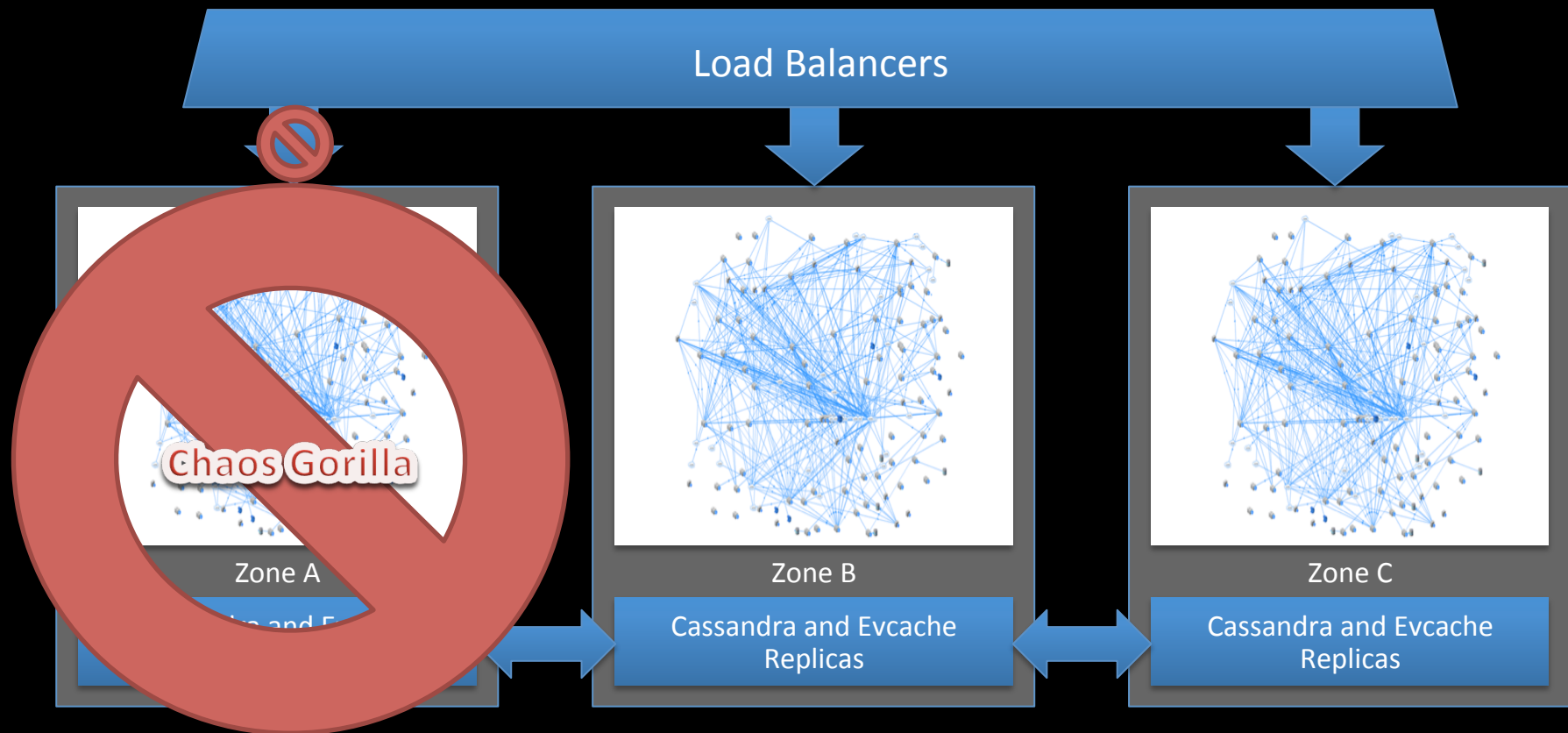
# Component Micro-Services

Test With Chaos Monkey, Latency Monkey



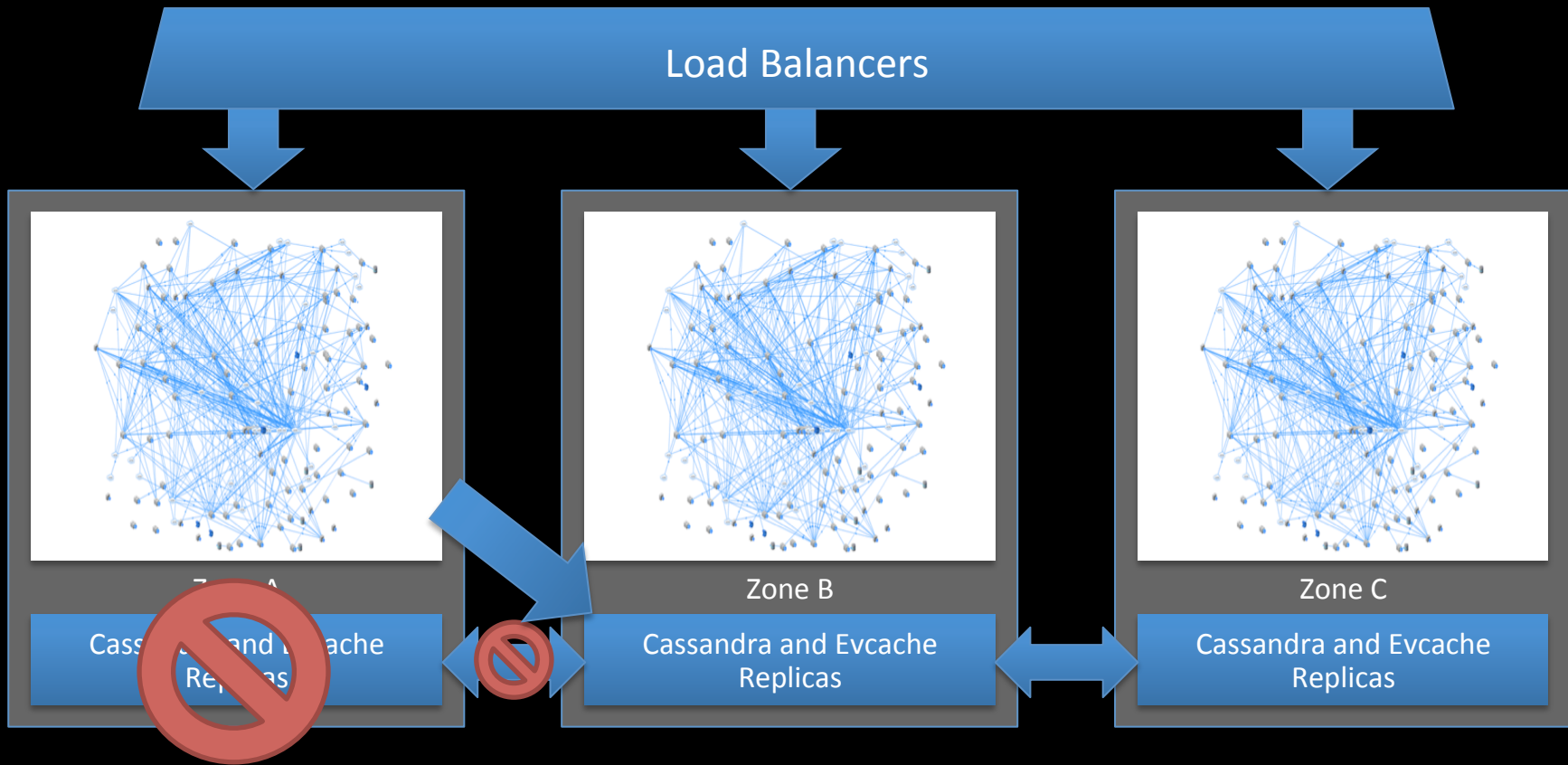
# Three Balanced Availability Zones

Test with Chaos Gorilla

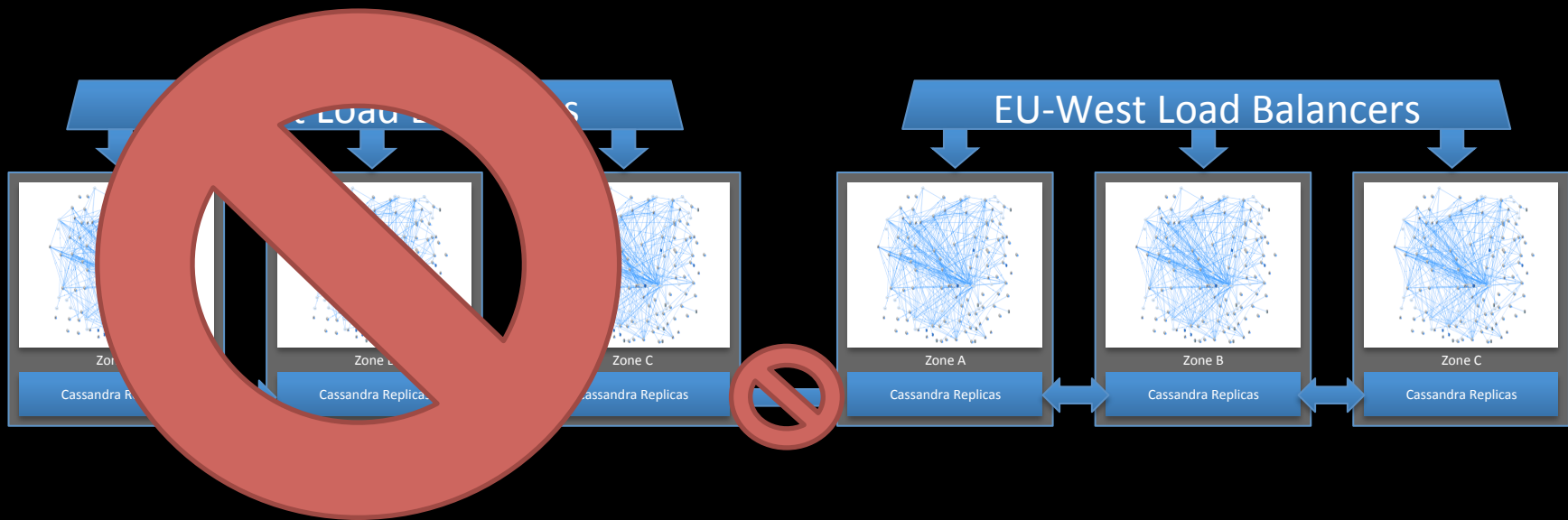


# Triple Replicated Persistence

Cassandra maintenance affects individual replicas



# Isolated Regions





# Failure Modes and Effects

Failure Mode	Probability	Current Mitigation Plan
Application Failure	High	Automatic degraded response
AWS Region Failure	Low	Switch traffic between regions
AWS Zone Failure	Medium	Continue to run on 2 out of 3 zones
Datacenter Failure	Medium	Migrate more functions to cloud
Data store failure	Low	Restore from S3 backups
S3 failure	Low	Restore from remote archive

Until we got really good at mitigating high and medium probability failures, the ROI for mitigating regional failures didn't make sense. Working on it now.

# Antifragile Testing

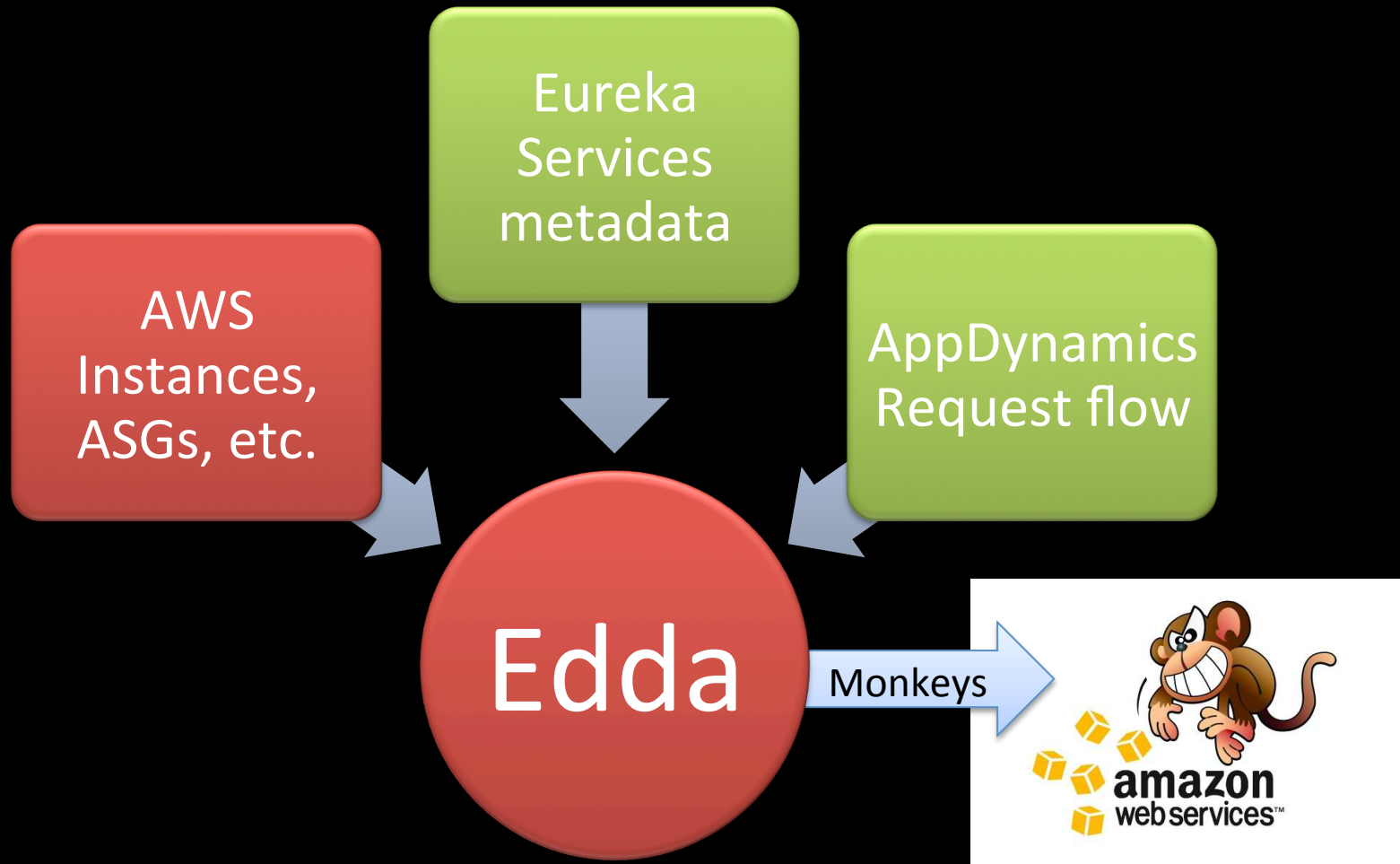
<http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html>

- Chaos Monkey makes sure systems are resilient
  - Kill individual instances without customer impact
- Chaos Gorilla shuts down entire zone
  - Run in production once every 3 months
- Latency Monkey
  - Injects extra latency and error return codes



# Edda – Configuration History

<http://techblog.netflix.com/2012/11/edda-learn-stories-of-your-cloud.html>



# Edda Query Examples

Find any instances that have ever had a specific public IP address

```
$ curl "http://edda/api/v2/view/instances;publicIpAddress=1.2.3.4;_since=0"  
["i-0123456789", "i-012345678a", "i-012345678b"]
```

Show the most recent change to a security group

```
$ curl "http://edda/api/v2/aws/securityGroups/sg-0123456789;_diff;_all;_limit=2"  
--- /api/v2/aws.securityGroups/sg-0123456789;_pp;_at=1351040779810  
+++ /api/v2/aws.securityGroups/sg-0123456789;_pp;_at=1351044093504  
@@ -1,33 +1,33 @@  
  {  
  ...  
    "ipRanges" : [  
      "10.10.1.1/32",  
      "10.10.1.2/32",  
+     "10.10.1.3/32",  
-     "10.10.1.4/32"  
    ...  
  }  
}
```

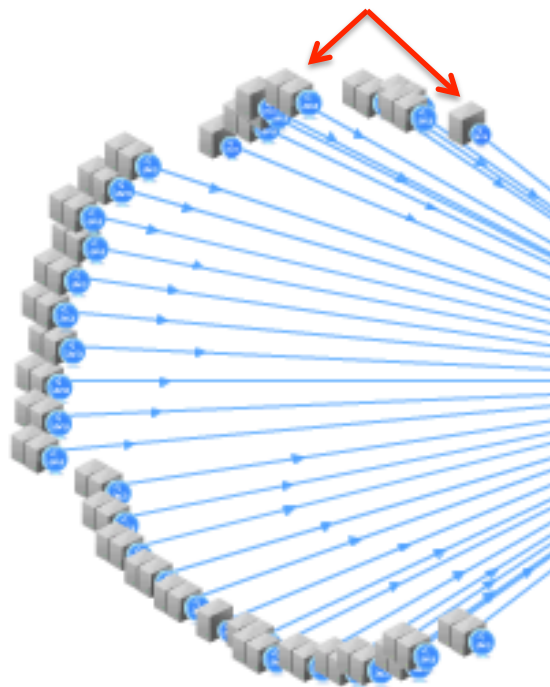
# Highly Available Storage

A highly scalable, available and durable deployment pattern based on Apache Cassandra

# Single Function Micro-Service Pattern

One keyspace, replaces a single table or materialized view

Many Different Single-Function REST Clients



Stateless Data Access REST Service  
Astyanax Cassandra Client

Single function Cassandra  
Cluster Managed by Priam  
Between 6 and 144 nodes

Over 50 Cassandra clusters  
Over 1000 nodes  
Over 30TB backup  
Over 1M writes/s/cluster

Optional  
Datacenter  
Update Flow

Each icon represents a horizontally scaled service of three to hundreds of instances deployed over three availability zones

# Stateless Micro-Service Architecture

## Linux Base AMI (CentOS or Ubuntu)

Optional  
Apache  
frontend,  
memcached,  
non-java apps

Monitoring  
Log rotation  
to S3  
AppDynamics  
machineagent  
Epic/Atlas

## Java (JDK 6 or 7)

AppDynamics  
appagent  
monitoring

GC and thread  
dump logging

## Tomcat

Application war file, base  
servlet, platform, client  
interface jars, Astyanax

Healthcheck, status  
servlets, JMX interface,  
Servo autoscale

# Cassandra Instance Architecture

Linux Base AMI (CentOS or Ubuntu)

Tomcat and  
Priam on JDK  
Healthcheck,  
Status

Monitoring  
AppDynamics  
machineagent  
Epic/Atlas

## Java (JDK 7)

AppDynamics  
appagent  
monitoring

GC and thread  
dump logging

## Cassandra Server

Local Ephemeral Disk Space – 2TB of SSD or 1.6TB disk  
holding Commit log and SSTables



# Priam – Cassandra Automation

Available at <http://github.com/netflix>

- Netflix Platform Tomcat Code
- Zero touch auto-configuration
- State management for Cassandra JVM
- Token allocation and assignment
- Broken node auto-replacement
- Full and incremental backup to S3
- Restore sequencing from S3
- Grow/Shrink Cassandra “ring”

# ETL for Cassandra

- Data is de-normalized over many clusters!
- Too many to restore from backups for ETL
- Solution – read backup files using Hadoop
- Aegisthus
  - <http://techblog.netflix.com/2012/02/aegisthus-bulk-data-pipeline-out-of.html>
  - High throughput raw SSTable processing
  - Re-normalizes many clusters to a consistent view
  - Extract, Transform, then Load into Teradata

# Cloud Native Big Data

Size the cluster to the data

Size the cluster to the questions

Never wait for space or answers

NETFLIX

OSS

# Netflix Dataoven

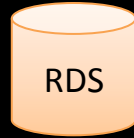
From cloud Services  
~100 Billion Events/day



From C\*  
Terabytes of Dimension data



Data Pipelines



Metadata



Data Warehouse  
Over 2 Petabytes

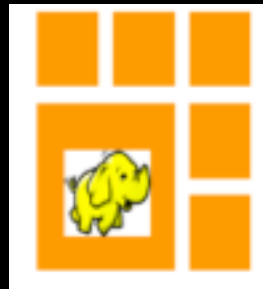


Gateways

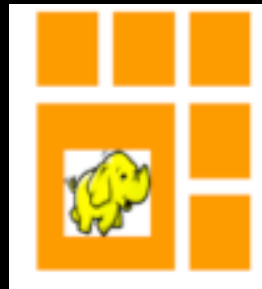


Tools

Hadoop Clusters – AWS EMR



1300 nodes



800 nodes



Multiple 150 nodes Nightly

# Global Architecture

Local Client Traffic to Cassandra

Synchronous Replication Across Zones

Asynchronous Replication Across Regions

# Astyanax Cassandra Client for Java

Available at <http://github.com/netflix>

- Features
  - Complete abstraction of connection pool from RPC protocol
  - Fluent Style API
  - Operation retry with backoff
  - Token aware
- Recipes
  - Distributed row lock (without zookeeper)
  - Multi-region row lock
  - Uniqueness constraint
  - Multi-row uniqueness constraint
  - Chunked and multi-threaded large file storage
  - Reverse index search
  - All rows query
  - Durable message queue

# Astyanax - Cassandra Write Data Flows

Single Region, Multiple Availability Zone, Token Aware

1. Client Writes to local coordinator
2. Coordinator writes to other zones
3. Nodes return ack
4. Data written to internal commit log disks (no more than 10 seconds later)



If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

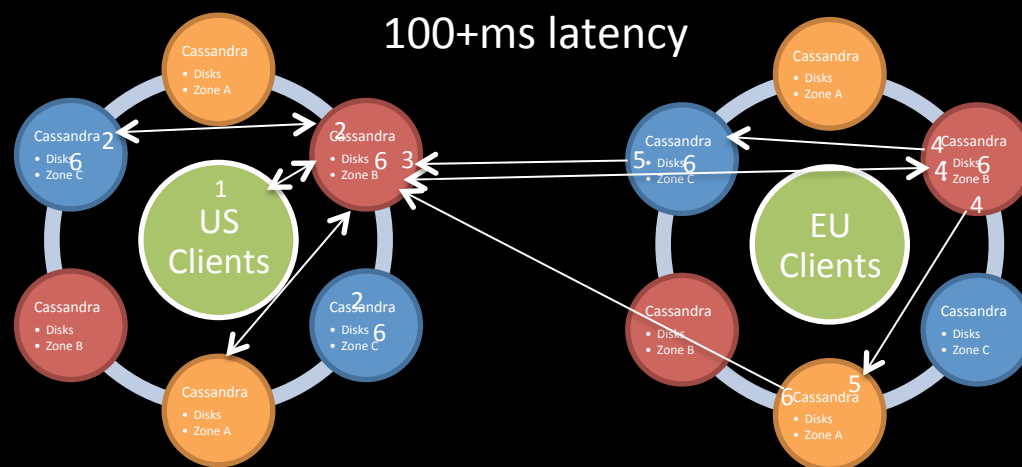
SSTable disk writes and compactions occur asynchronously

# Data Flows for Multi-Region Writes

## Token Aware, Consistency Level = Local Quorum

1. Client writes to local replicas
2. Local write acks returned to Client which continues when 2 of 3 local nodes are committed
3. Local coordinator writes to remote coordinator.
4. When data arrives, remote coordinator node acks and copies to other remote zones
5. Remote nodes ack to local coordinator
6. Data flushed to internal commit log disks (no more than 10 seconds later)

If a node or region goes offline, hinted handoff completes the write when the node comes back up. Nightly global compare and repair jobs ensure everything stays consistent.





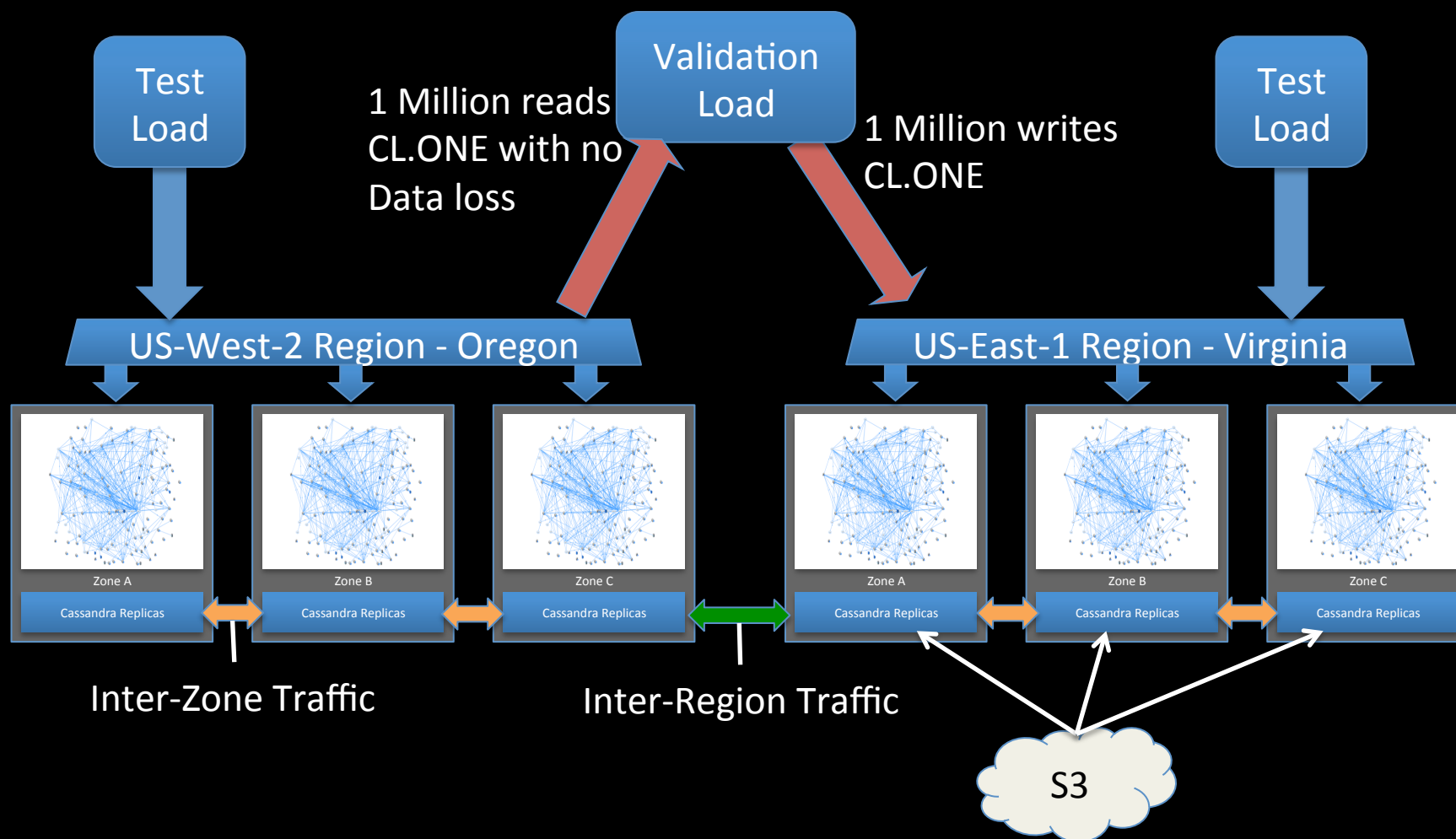
# Cross Region Use Cases

- Geographic Isolation
  - US to Europe replication of subscriber data
  - Read intensive, low update rate
  - Production use since late 2011
- Redundancy for regional failover
  - US East to US West replication of everything
  - Includes write intensive data, high update rate
  - Testing now

# Benchmarking Global Cassandra

Write intensive test of cross region capacity

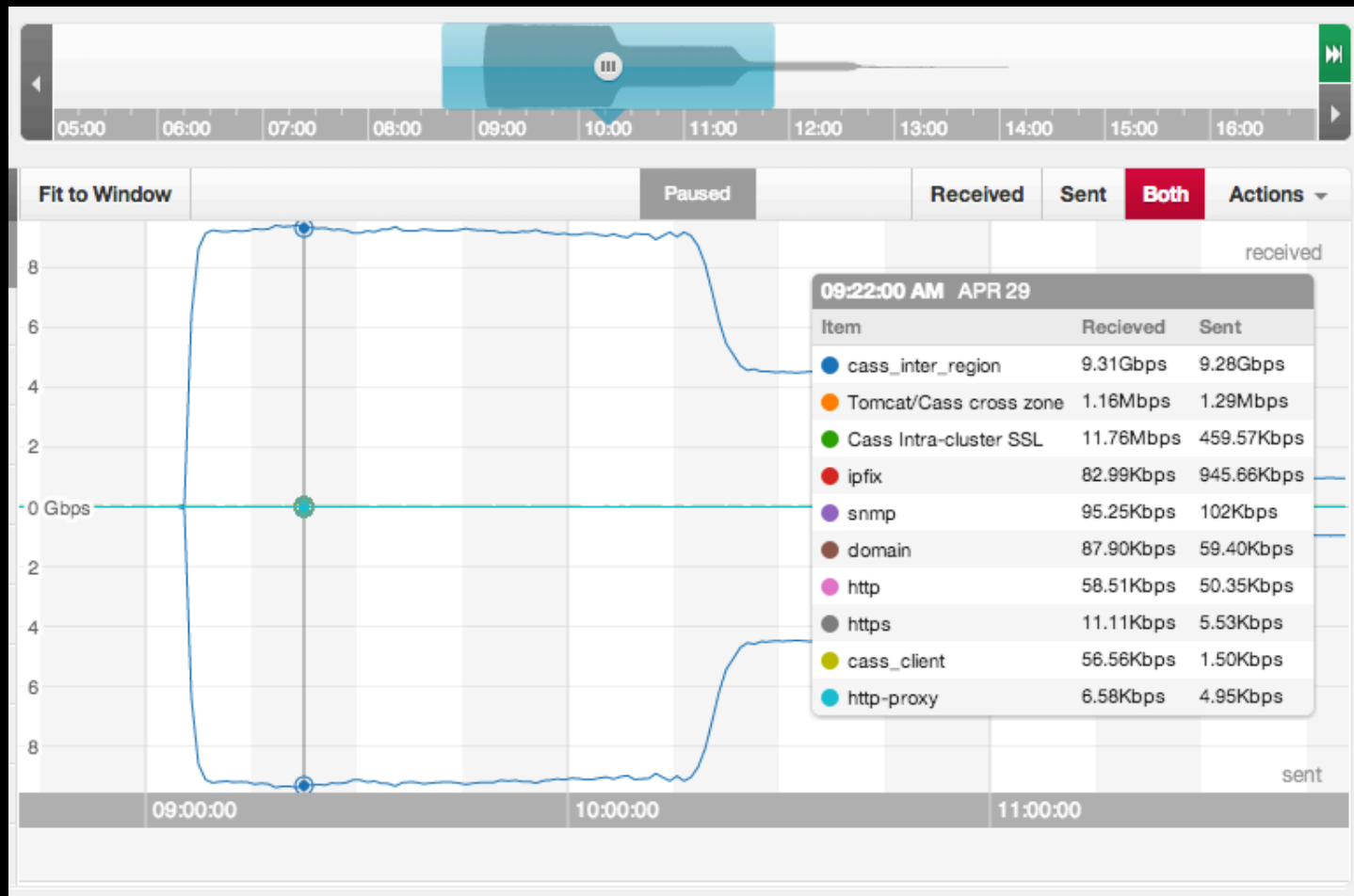
16 x hi1.4xlarge SSD nodes per zone = 96 total



# Copying 18TB from East to West

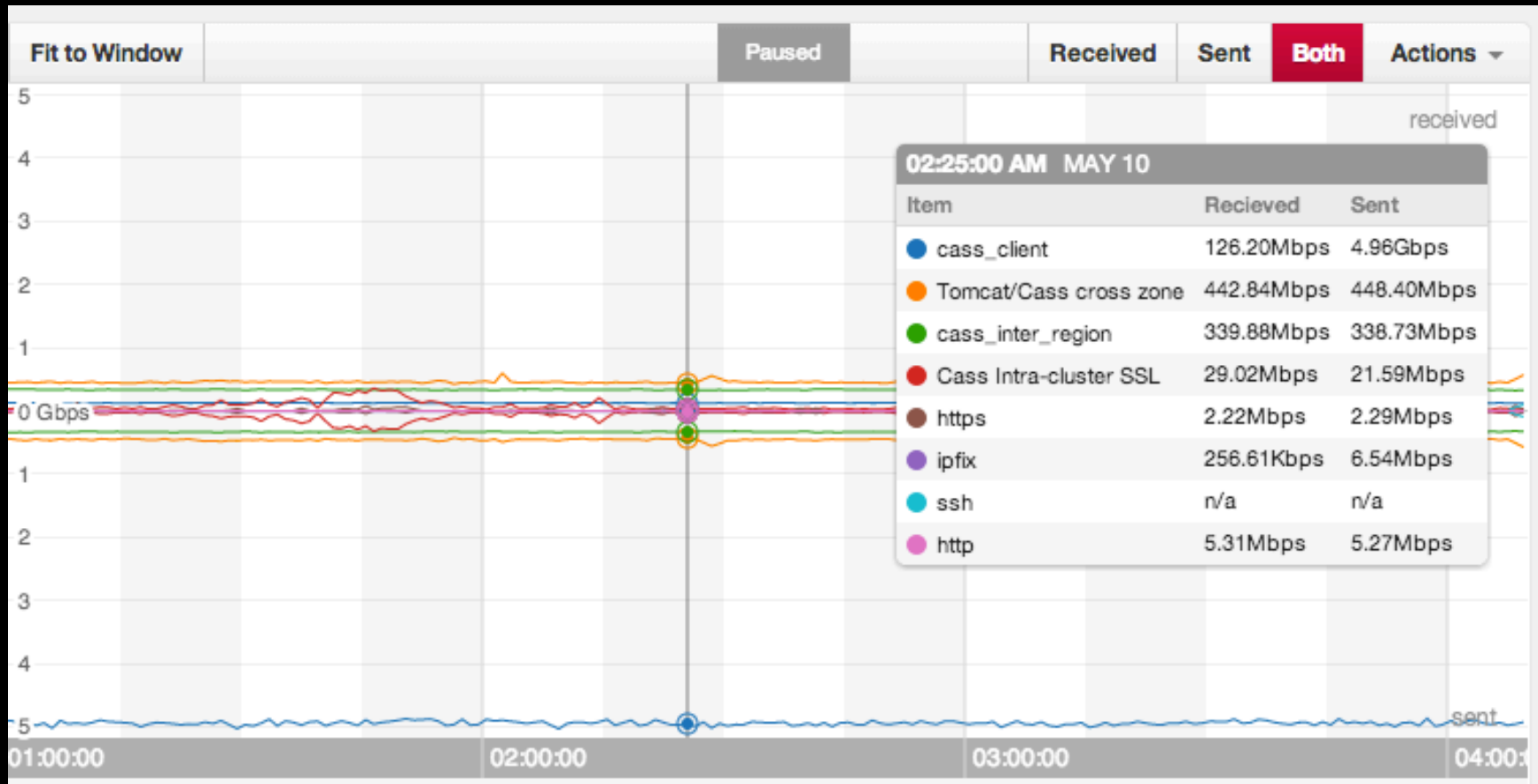
Cassandra bootstrap 9.3 Gbit/s single threaded 48 nodes to 48 nodes

Thanks to boundary.com for these network analysis plots



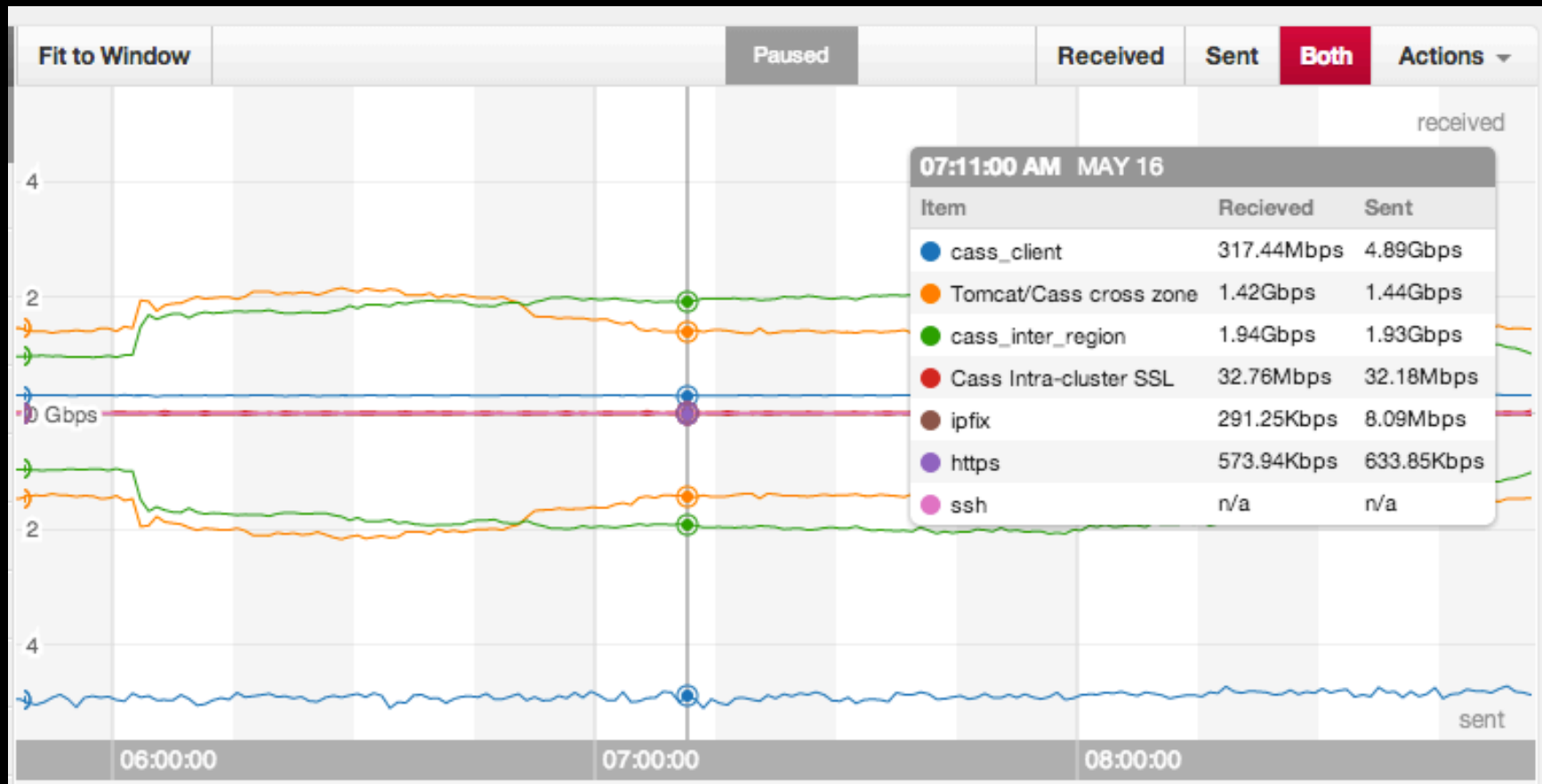
# Inter Region Traffic Test

Verified at desired capacity, no problems, 339 MB/s, 83ms latency

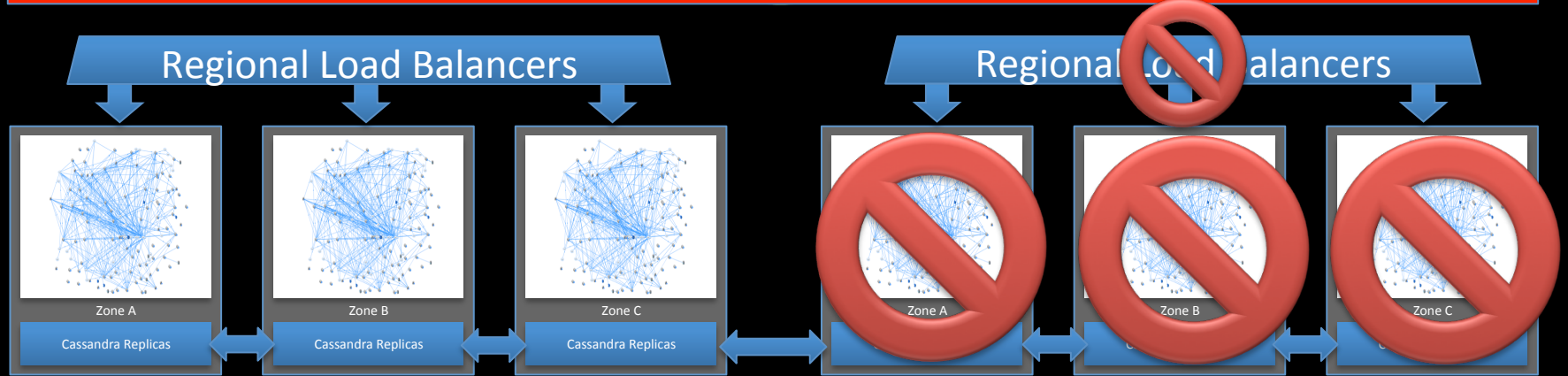
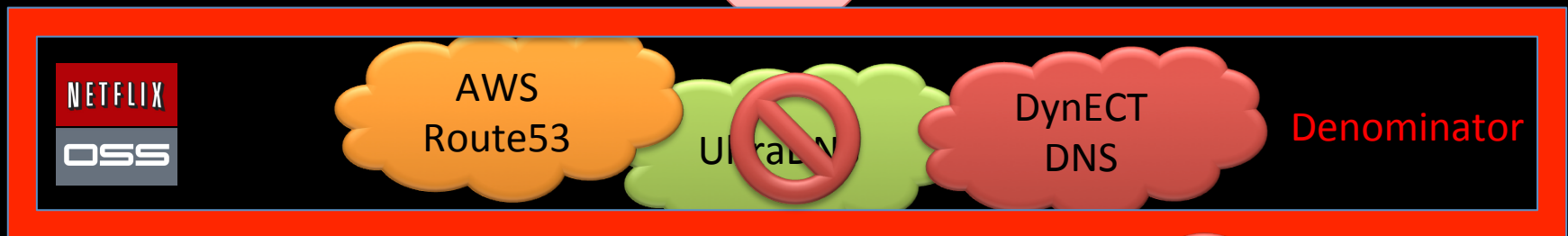


# Ramp Up Load Until It Breaks!

Unmodified tuning, dropping client data at 1.93GB/s inter region traffic  
Spare CPU, IOPS, Network, just need some Cassandra tuning for more



# Managing Multi-Region Availability



Denominator – manage traffic via multiple DNS providers



Boosting the @NetflixOSS Ecosystem

See [netflix.github.com](https://netflix.github.com)

**Judges choice award**

**Best example application mash-up**

**Best usability enhancement**

**Best portability enhancement**

**Best new monkey**

**Best new feature**

**Best datastore integration**

**Best contribution to code quality**

**Best contribution to operational tools**

**Best contribution to performance**





Aino Corry  
Program Chair for Qcon/GOTO



Simon Wardley  
Strategist



Martin Fowler  
Chief Scientist Thoughtworks



Werner Vogels  
CTO Amazon



Joe Weinman  
SVP Telx, Author "Cloudeconomics"



Yury Izrailevsky  
VP Cloud Netflix

# What do you win?

One winner in each of the 10 categories

Ticket and expenses to attend AWS

Re:Invent 2013 in Las Vegas

A Trophy

**\$10,000 cash and \$5,000 in AWS  
Credits**



A Cloud Native Open Source Platform

See [netflix.github.com](https://netflix.github.com)

# Netflix Platform Evolution

2009-2010

Bleeding Edge  
Innovation

2011-2012

Common  
Pattern

2013-2014

Shared  
Pattern

Netflix ended up several years ahead of the industry, but it's becoming commoditized now

Establish our  
solutions as Best  
Practices / Standards

Hire, Retain and  
Engage Top  
Engineers

Goals

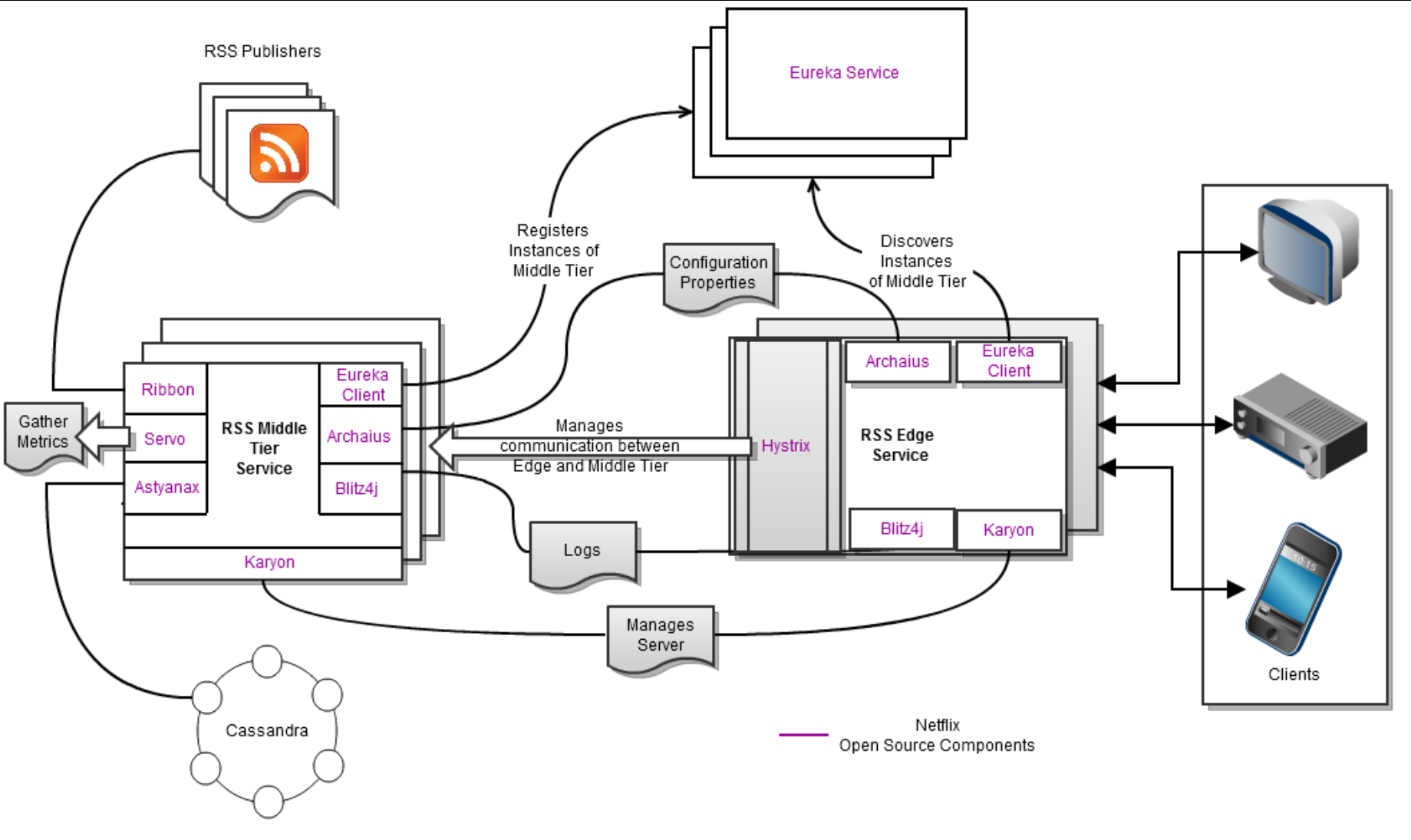
Build up Netflix  
Technology Brand

Benefit from a  
shared ecosystem

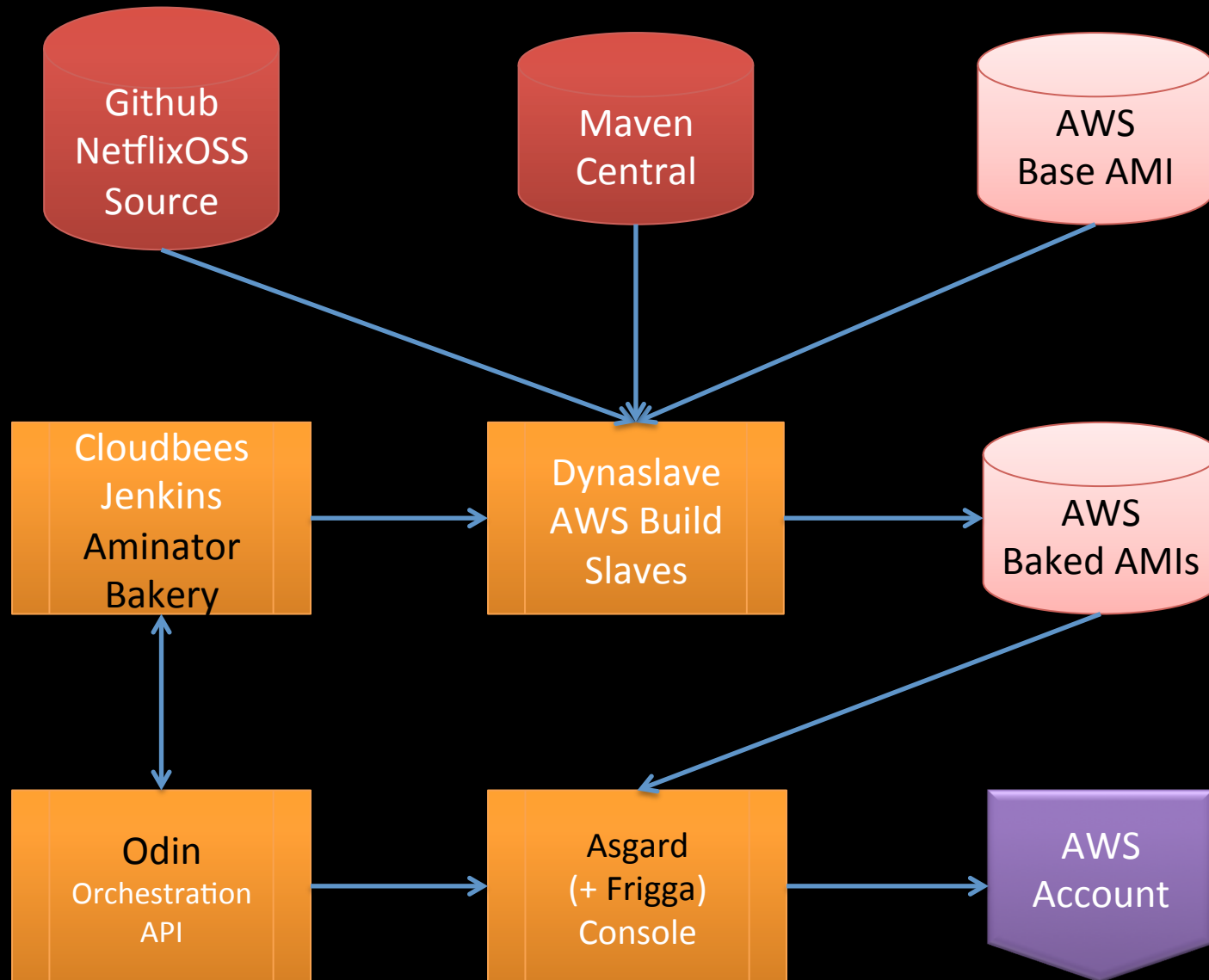
How does it all fit together?



# Example Application – RSS Reader



# NetflixOSS Continuous Build and Deployment





# NetflixOSS Services Scope

## AWS Account

Asgard Console

Archaius  
Config Service

Cross region  
Priam C\*

Pytheas  
Dashboards

Atlas  
Monitoring

Genie, Lipstick  
Hadoop Services

AWS Usage  
Cost Monitoring

## Multiple AWS Regions

Eureka Registry

Exhibitor ZK

Edda History

Simian Army

Zuul Traffic Mgr

## 3 AWS Zones

Application  
Clusters  
Autoscale Groups  
Instances

Priam  
Cassandra  
Persistent Storage

Evcache  
Memcached  
Ephemeral Storage

# NetflixOSS Instance Libraries

## Initialization

- Baked AMI – Tomcat, Apache, your code
- Governor – Guice based dependency injection
- Archaius – dynamic configuration properties client
- Eureka - service registration client

## Service Requests

- Karyon - Base Server for inbound requests
- RxJava – Reactive pattern
- Hystrix/Turbine – dependencies and real-time status
- Ribbon - REST Client for outbound calls

## Data Access

- Astyanax – Cassandra client and pattern library
- Evcache – Zone aware Memcached client
- Curator – Zookeeper patterns
- Denominator – DNS routing abstraction

## Logging

- Blitz4j – non-blocking logging
- Servo – metrics export for autoscaling
- Atlas – high volume instrumentation

# Dashboards with Pytheas (Explorers)

<http://techblog.netflix.com/2013/05/announcing-pytheas.html>

- Cassandra Explorer
  - Browse clusters, keyspaces, column families
- Base Server Explorer
  - Browse service endpoints configuration, perf
- Anything else you want to build...

# Cassandra Explorer

CASSANDRA EXPLORER: -

Region: test.eu-west-1 [Explorer](#) | [Admin](#) | [Dashboard](#)

Filter: 0 Refresh [Home](#)

## Cluster: CASS\_SANDBOX

**Keyspace: CacheContent**

StrategyClass org.apache.cassandra.locator.NetworkTopologyStrategy  
us-east 3  
Column Families: [RegionalManifestIndex](#) [ReportedManifest](#) [RegionalManifest](#)

**Keyspace: vault2**

StrategyClass org.apache.cassandra.locator.NetworkTopologyStrategy  
us-east 3  
Column Families: [mopstore](#)

**Keyspace: AstyanaxUnitTests**

StrategyClass org.apache.cassandra.locator.SimpleStrategy  
replication\_factor 3  
Column Families: [CompositeKev](#) [LonaColumn1](#) [users](#) [CompositeColumn](#) [ClickStream](#) [Standard2](#) [Standard1](#)  
[CompositeCsv](#) [TimeUUID1](#) [Counter1](#)

**Keyspace: trackid\_Keyspace**

StrategyClass org.apache.cassandra.locator.NetworkTopologyStrategy  
us-east 3  
Column Families: [trackIds](#)

**Keyspace: vault**

- ABCASSANDRA
- CASS\_ABMR\_2
- CASS\_API\_MULTIREGION
  - CASS\_API\_TEST
- CASS\_BIG\_SCALE
- CASS\_BOOTSTRAP
- CASS\_BRISK\_TEST
- CASS\_BULKPREDICTION
- CASS\_CCS
- CASS\_CDE
- CASS\_DMS
- CASS\_ECOMM
- CASS\_GENPOP
- CASS\_GPSAPPLICATION\_US
- CASS\_MERCH
- CASS\_MRGENPOP
- CASS\_MRSUB
- CASS\_MRTEST
- CASS\_NCCP\_US
- CASS\_ORESTES
- CASS\_PBS\_US
- CASS\_PERF\_SUB
- CASS\_QAREG\_SUBSCRIBER
- CASS\_RTAL\_GPS
- CASS\_SANDBOX
  - CacheContent
  - vault2
  - AstyanaxUnitTests
  - trackid\_Keyspace
  - vault
  - video\_presentations\_Keyspace
  - IntTest
- CASS\_SOCIAL
- CASS\_STREAM\_LATAM
- CASS\_STREAM\_MR
- CASS\_STREAM\_REFRESH
- CASS\_SUBSCRIBER
- CASS\_TEST\_JYOTI
- CASS\_TRACERS
- CASS\_TURTLE
- CASS\_VMS

# Cassandra Explorer

CASSANDRA EXPLORER: test-us-east-1

Region: test-us-east-1 Column Family: CASS\_SANDBOX.AstyanaxUnitTests.TimeUUID1 Filter: sand 29

[Explorer](#) [Admin](#) [Dashboard](#)

Key: UTF8Type limit 100 Column Range TimeUUIDType => TimeUUIDType Limit: 100 TimeUUIDType Execute

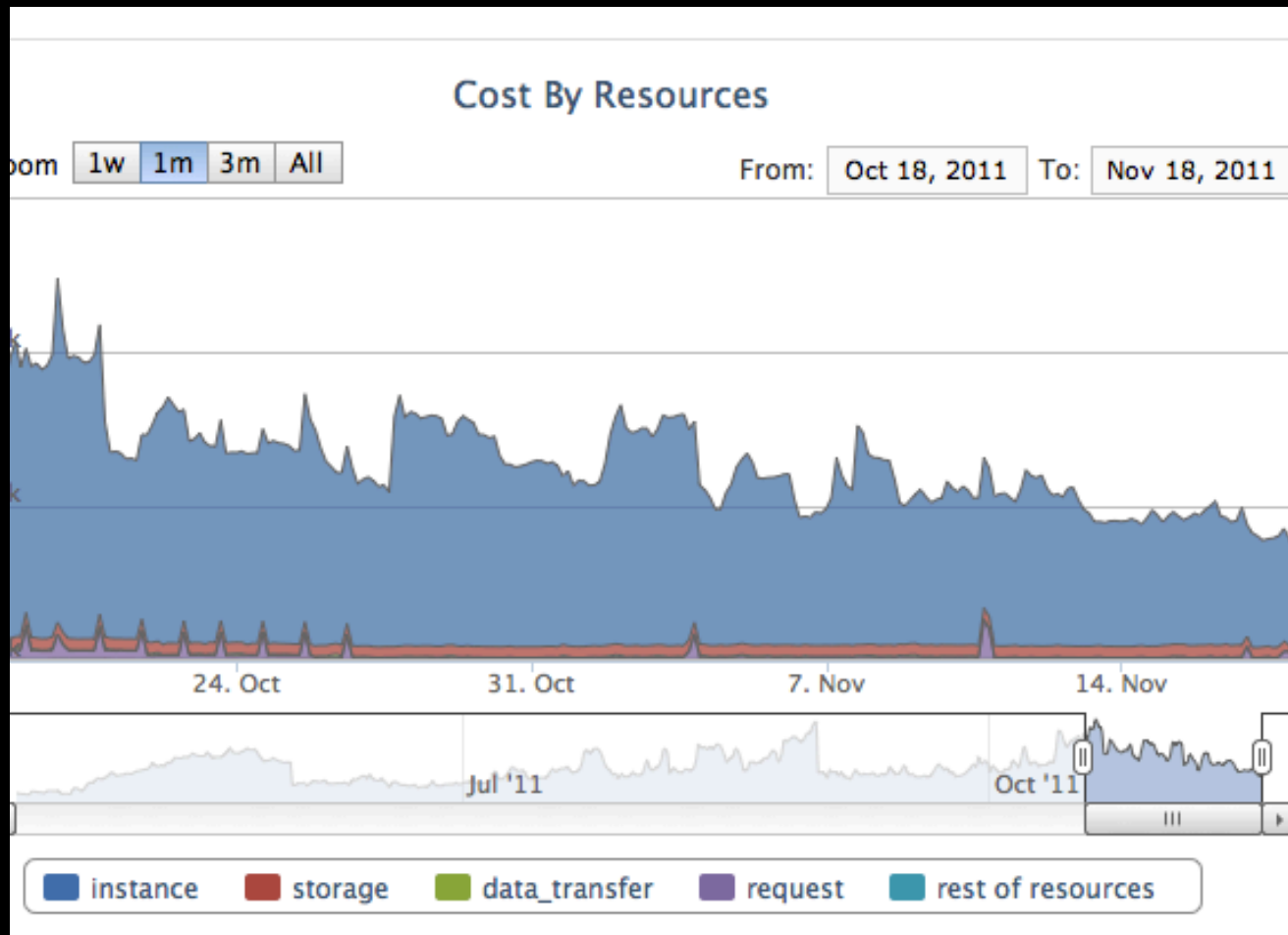
key	column	value	timestamp
Key1	2011-11-15T19:21:18.730+0000	0000002a	2011-11-15T19:21:18.830+0000
Key1	2011-11-15T19:21:18.830+0000	00000064	2011-11-15T19:21:18.830+0000
Key1	2011-11-15T19:21:18.831+0000	00000065	2011-11-15T19:21:18.830+0000
Key1	2011-11-15T19:21:18.832+0000	00000066	2011-11-15T19:21:18.830+0000
Key1	2011-11-15T19:21:18.833+0000	00000067	2011-11-15T19:21:18.830+0000
Key1	2011-11-15T19:21:18.834+0000	00000068	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.835+0000	00000069	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.836+0000	0000006a	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.837+0000	0000006b	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.838+0000	0000006c	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.839+0000	0000006d	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.840+0000	0000006e	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.841+0000	0000006f	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.842+0000	00000070	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.843+0000	00000071	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.844+0000	00000072	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.845+0000	00000073	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.846+0000	00000074	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.847+0000	00000075	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.848+0000	00000076	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.849+0000	00000077	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.850+0000	00000078	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.851+0000	00000079	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.852+0000	0000007a	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.853+0000	0000007b	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.854+0000	0000007c	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.855+0000	0000007d	2011-11-15T19:21:18.831+0000
Key1	2011-11-15T19:21:18.856+0000	0000007e	2011-11-15T19:21:18.831+0000

Error: Host: ec2-204-236-213-136.compute-1.amazonaws.com(10.218.23.152):7102 Duration: 8.313 Count: 200

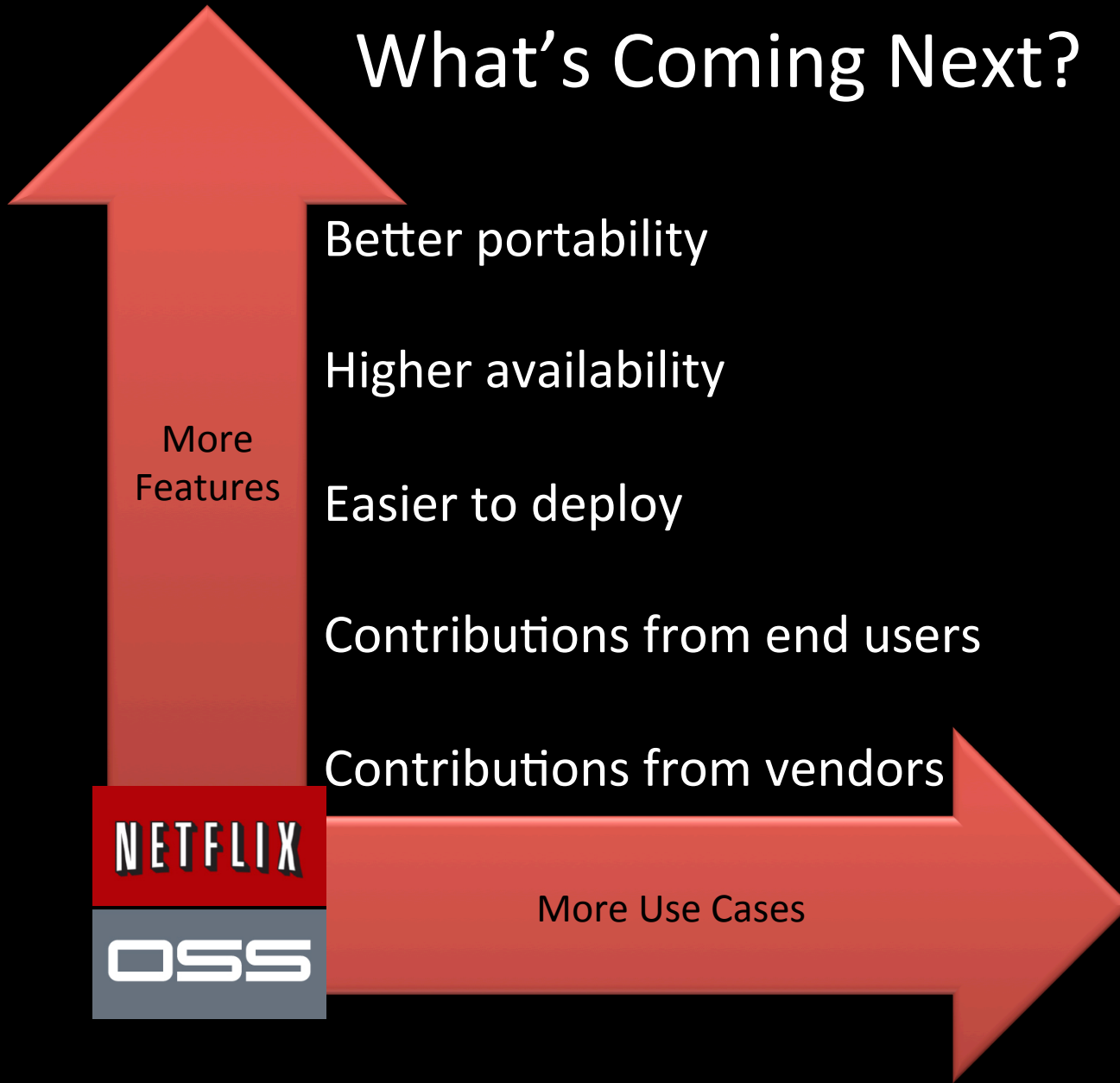


# AWS Usage (coming soon)

Reservation-aware cost monitoring and reporting



# What's Coming Next?





**NETFLIX**

**OSS**

OPEN SOURCE SOFTWARE

Functionality and scale now, portability coming

Moving from parts to a platform in 2013

Netflix is fostering a cloud native ecosystem

**Rapid Evolution - Low MTBIAMSH**

(Mean Time Between Idea And Making Stuff Happen)

# Takeaway

*NetflixOSS makes it easier for everyone to become Cloud Native*

<http://netflix.github.com>

<http://techblog.netflix.com>

<http://slideshare.net/Netflix>

<http://www.linkedin.com/in/adriancockcroft>

@adrianco #netflixcloud @NetflixOSS



# Slideshare NetflixOSS Details

- Lightning Talks Feb S1E1
  - <http://www.slideshare.net/RuslanMeshenberg/netflixoss-open-house-lightning-talks>
- Asgard In Depth Feb S1E1
  - <http://www.slideshare.net/joesondow/asgard-overview-from-netflix-oss-open-house>
- Lightning Talks March S1E2
  - <http://www.slideshare.net/RuslanMeshenberg/netflixoss-meetup-lightning-talks-and-roadmap>
- Security Architecture
  - [http://www.slideshare.net/jason\\_chan/](http://www.slideshare.net/jason_chan/)
- Cost Aware Cloud Architectures – with Jinesh Varia of AWS
  - <http://www.slideshare.net/AmazonWebServices/building-costaware-architectures-jinesh-varia-aws-and-adrian-cockroft-netflix>

# Amazon Cloud Terminology Reference

See <http://aws.amazon.com/> This is not a full list of Amazon Web Service features

- AWS – Amazon Web Services (common name for Amazon cloud)
- AMI – Amazon Machine Image (archived boot disk, Linux, Windows etc. plus application code)
- EC2 – Elastic Compute Cloud
  - Range of virtual machine types m1, m2, c1, cc, cg. Varying memory, CPU and disk configurations.
  - Instance – a running computer system. Ephemeral, when it is de-allocated nothing is kept.
  - Reserved Instances – pre-paid to reduce cost for long term usage
  - Availability Zone – datacenter with own power and cooling hosting cloud instances
  - Region – group of Avail Zones – US-East, US-West, EU-Eire, Asia-Singapore, Asia-Japan, SA-Brazil, US-Gov
- ASG – Auto Scaling Group (instances booting from the same AMI)
- S3 – Simple Storage Service (http access)
- EBS – Elastic Block Storage (network disk filesystem can be mounted on an instance)
- RDS – Relational Database Service (managed MySQL master and slaves)
- DynamoDB/SDB – Simple Data Base (hosted http based NoSQL datastore, DynamoDB replaces SDB)
- SQS – Simple Queue Service (http based message queue)
- SNS – Simple Notification Service (http and email based topics and messages)
- EMR – Elastic Map Reduce (automatically managed Hadoop cluster)
- ELB – Elastic Load Balancer
- EIP – Elastic IP (stable IP address mapping assigned to instance or ELB)
- VPC – Virtual Private Cloud (single tenant, more flexible network and security constructs)
- DirectConnect – secure pipe from AWS VPC to external datacenter
- IAM – Identity and Access Management (fine grain role based security keys)

