Cloud Native, Capacity, Performance and Cost Optimization Tools and Techniques

CMG Workshop
November 2013

Adrian Cockcroft
@adrianco @NetflixOSS
http://www.linkedin.com/in/adriancockcroft

Presentation vs. Workshop

- Presentation
  - Short duration, focused subject
  - One presenter to many anonymous audience
  - A few questions at the end
- Workshop
  - Time to explore in and around the subject
  - Tutor gets to know the audience
  - Discussion, rat-holes, “bring out your dead”

Attendee Introductions

- Who are you, where do you work
- Why are you here today, what do you need
- “Bring out your dead”
  - Do you have a specific problem or question?
  - One sentence elevator pitch
- What instrument do you play?

Content

Cloud Native

Migration Path

Service and API Architectures

Storage Architecture

Operations and Tools

Cost Optimization

More?

Cloud Native

What is it?

Why?

Strive for perfection

Perfect code

Perfect hardware

Perfectly operated

But perfection takes too long…

Compromises…

Time to market vs. Quality

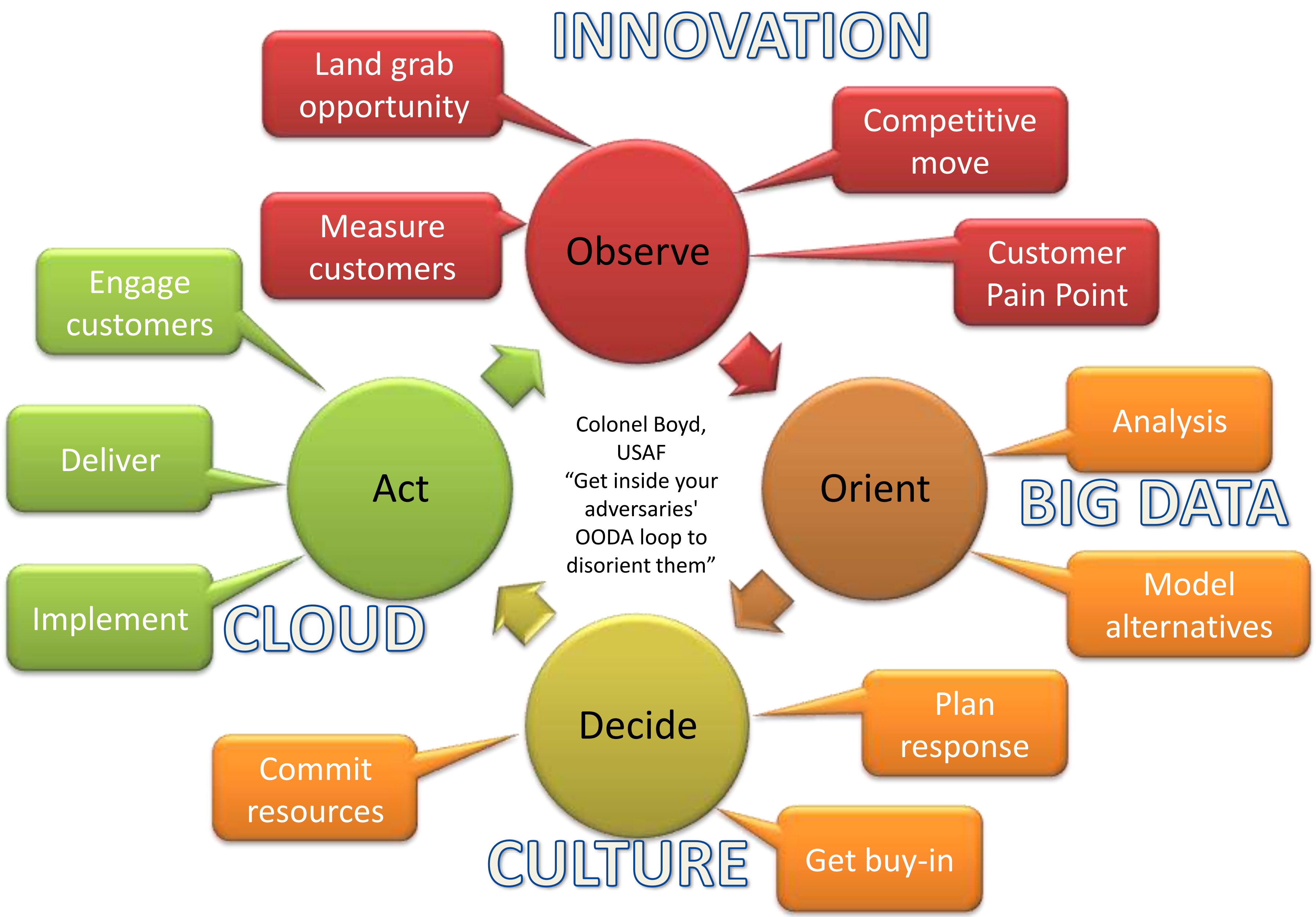
Utopia remains out of reach

Where time to market wins big

Making a land-grab

Disrupting competitors (OODA)

Anything delivered as web services


INNOVATION
Land grab opportunity
Competitive move
Measure customers
Observe
Customer Pain Point
Engage customers
Analysis
Deliver
Act
Colonel Boyd, USAF “Get inside your adversaries' OODA loop to disorient them”
Orient
BIG DATA
Model alternatives
Implement
CLOUD
Decide
Plan response
Commit resources
CULTURE
Get buy-in

How Soon?

Product features in days instead of months

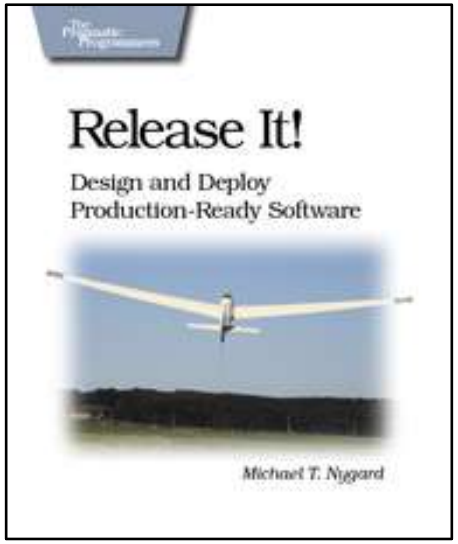
Deployment in minutes instead of weeks

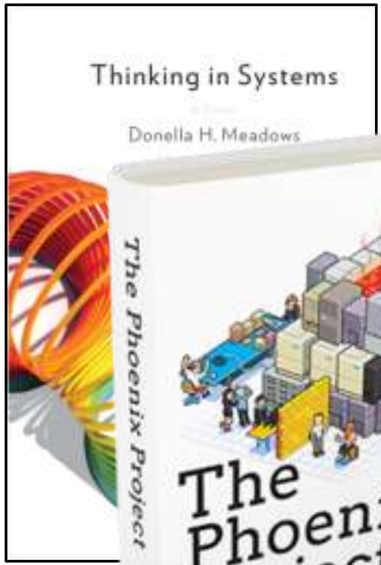
Incident response in seconds instead of hours

Cloud Native
A new engineering challenge

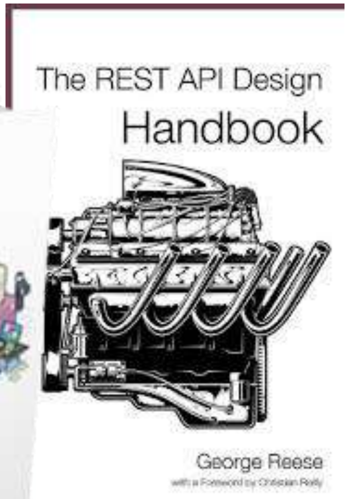
Construct a highly agile and highly available service from ephemeral and assumed broken components

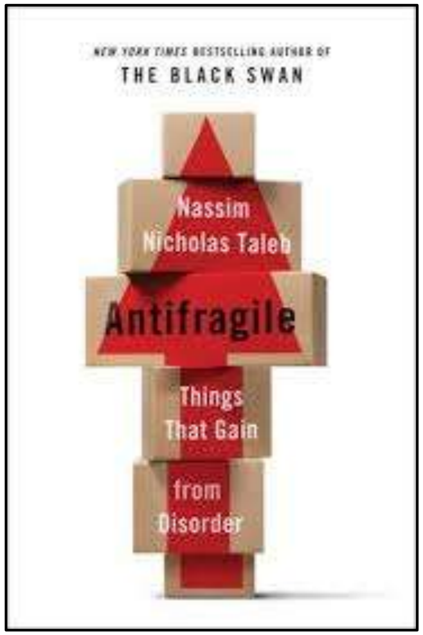
Inspiration

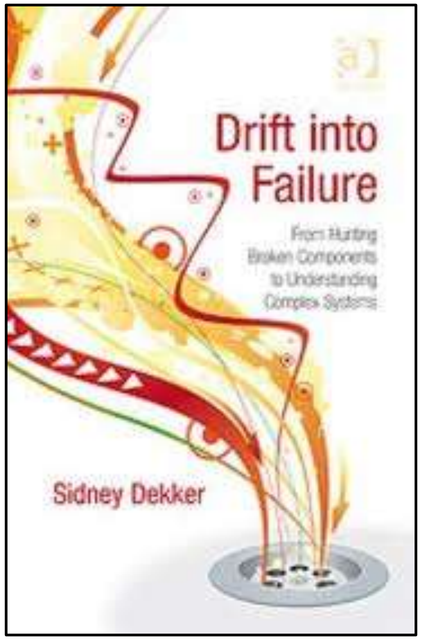


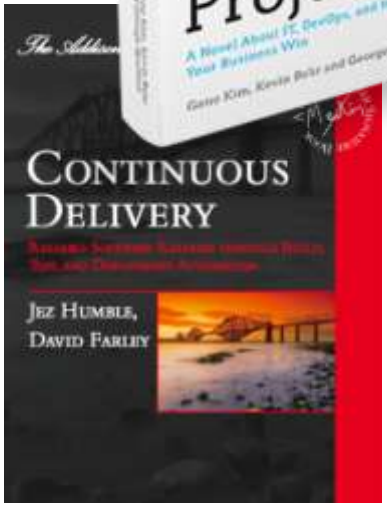


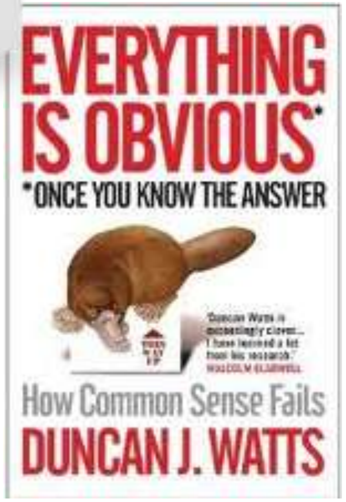
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr and George Spafford

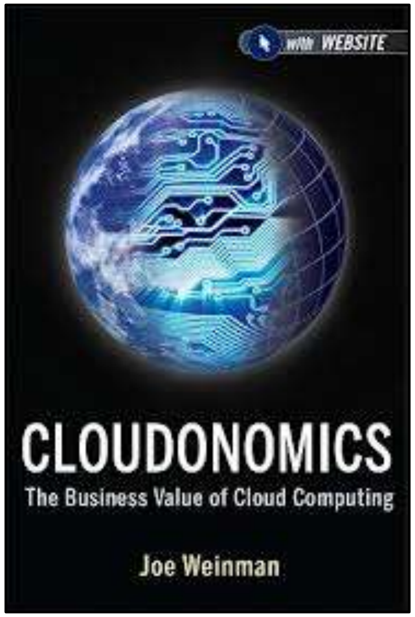



NEW YORK TIMES BESTSELLING AUTHOR OF THE BLACK SWAN
Nassim Nicholas Taleb
Antifragile
Things That Gain from Disorder


Drift into Failure
From Hunting Broken Components to Understanding Complex Systems
Sidney Dekker







How to get to Cloud Native

Freedom and Responsibility for Developers

Decentralize and Automate Ops Activities

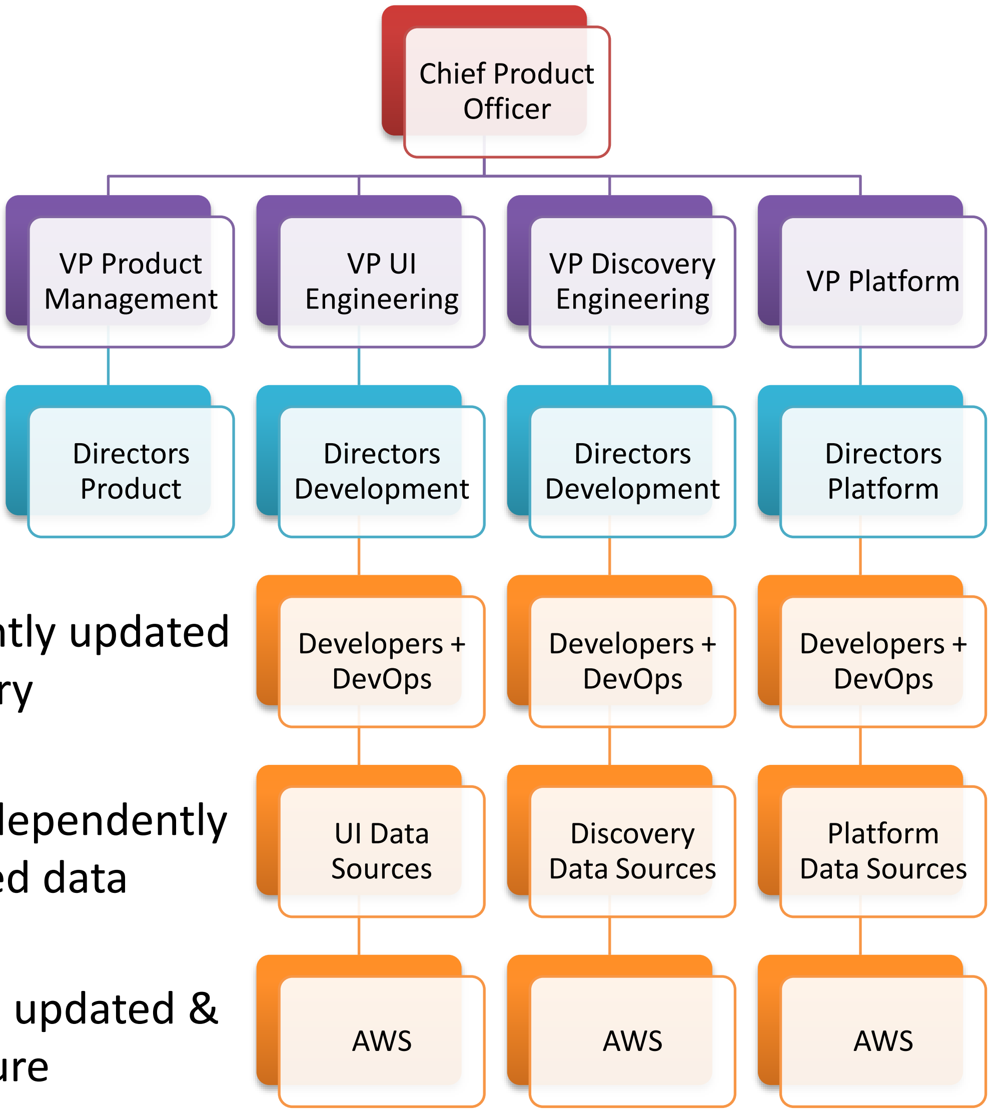
Integrate DevOps into the Business Organization

Re-Org!

Four Transitions

- Management: Integrated Roles in a Single Organization
  - Business, Development, Operations -> BusDevOps
- Developers: Denormalized Data – NoSQL
  - Decentralized, scalable, available, polyglot
- Responsibility from Ops to Dev: Continuous Delivery
  - Decentralized small daily production updates
- Responsibility from Ops to Dev: Agile Infrastructure - Cloud
  - Hardware in minutes, provisioned directly by developers

Netflix BusDevOps Organization



Code, independently updated continuous delivery

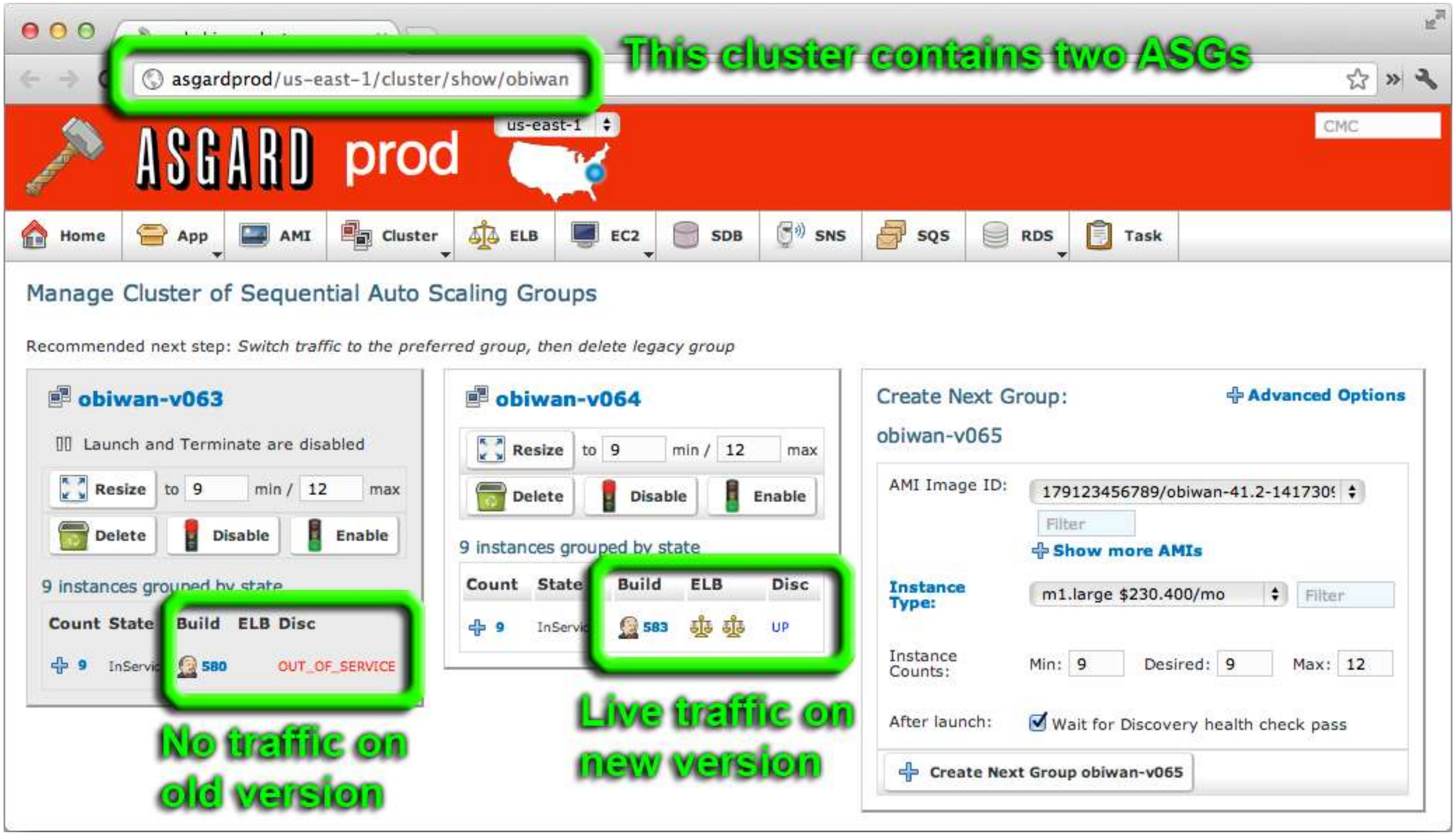
Denormalized, independently updated and scaled data

Cloud, self service updated & scaled infrastructure

Decentralized Deployment

Asgard Developer Portal

http://techblog.netflix.com/2012/06/asgard-web-based-cloud-management-and.html



This cluster contains two ASGs

asgardprod/us-east-1/cluster/show/obiwan

Manage Cluster of Sequential Auto Scaling Groups

Recommended next step: *Switch traffic to the preferred group, then delete legacy group*

obiwan-v063

Launch and Terminate are disabled

Resize to 9 min / 12 max

Delete | Disable | Enable

9 instances grouped by state

Count	State	Build	ELB	Disc
9	InServi	580		OUT_OF_SERVICE

obiwan-v064

Resize to 9 min / 12 max

Delete | Disable | Enable

9 instances grouped by state

Count	State	Build	ELB	Disc
9	InServi	583	⚖ ⚖	UP

Create Next Group: obiwan-v065

Advanced Options

AMI Image ID:	179123456789/obiwan-41.2-141730
	Filter
	Show more AMIs
Instance Type:	m1.large $230.400/mo
Instance Counts:	Min: 9 Desired: 9 Max: 12
After launch:	✓ Wait for Discovery health check pass

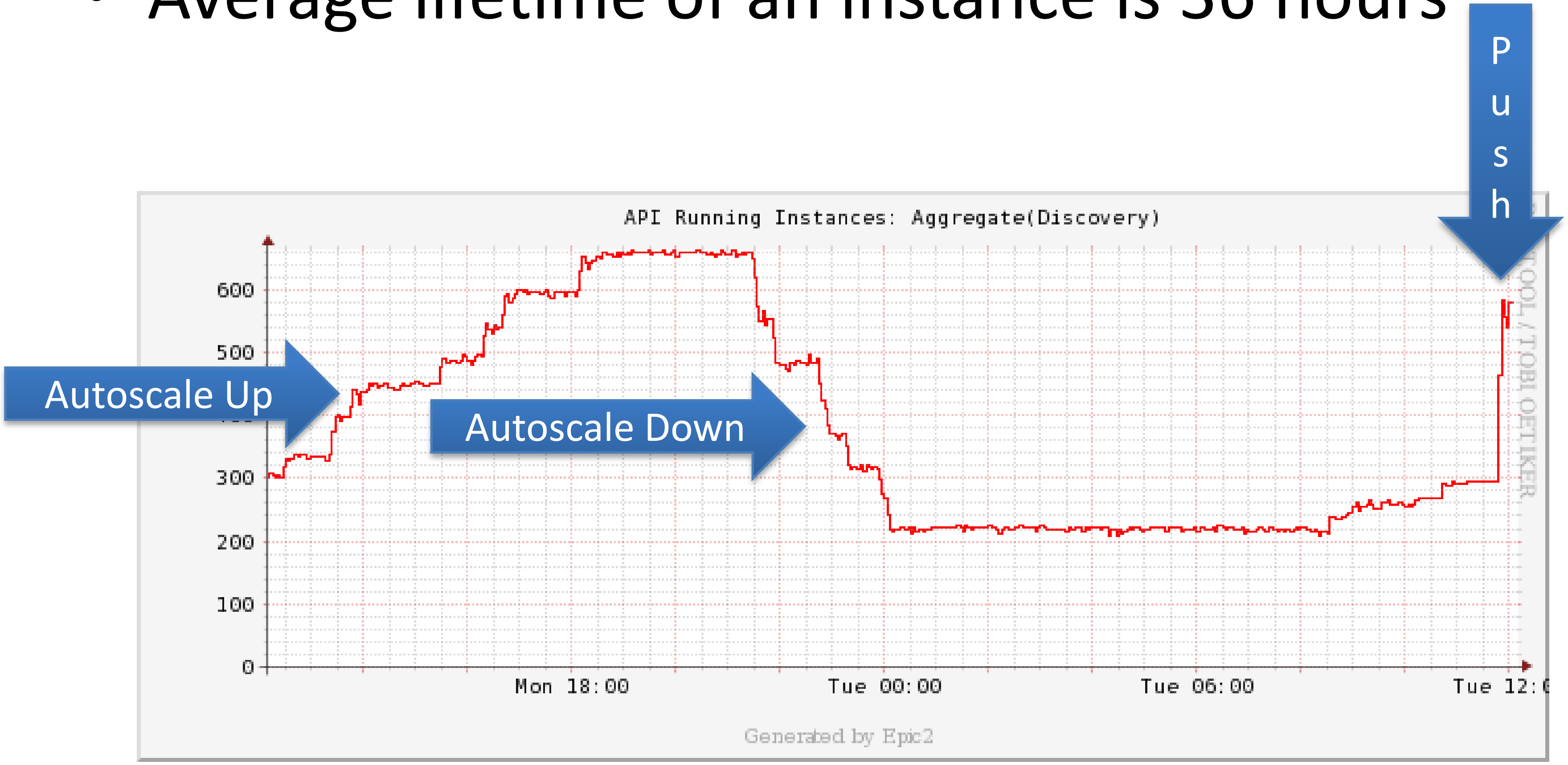
Create Next Group obiwan-v065

No traffic on old version

Live traffic on new version

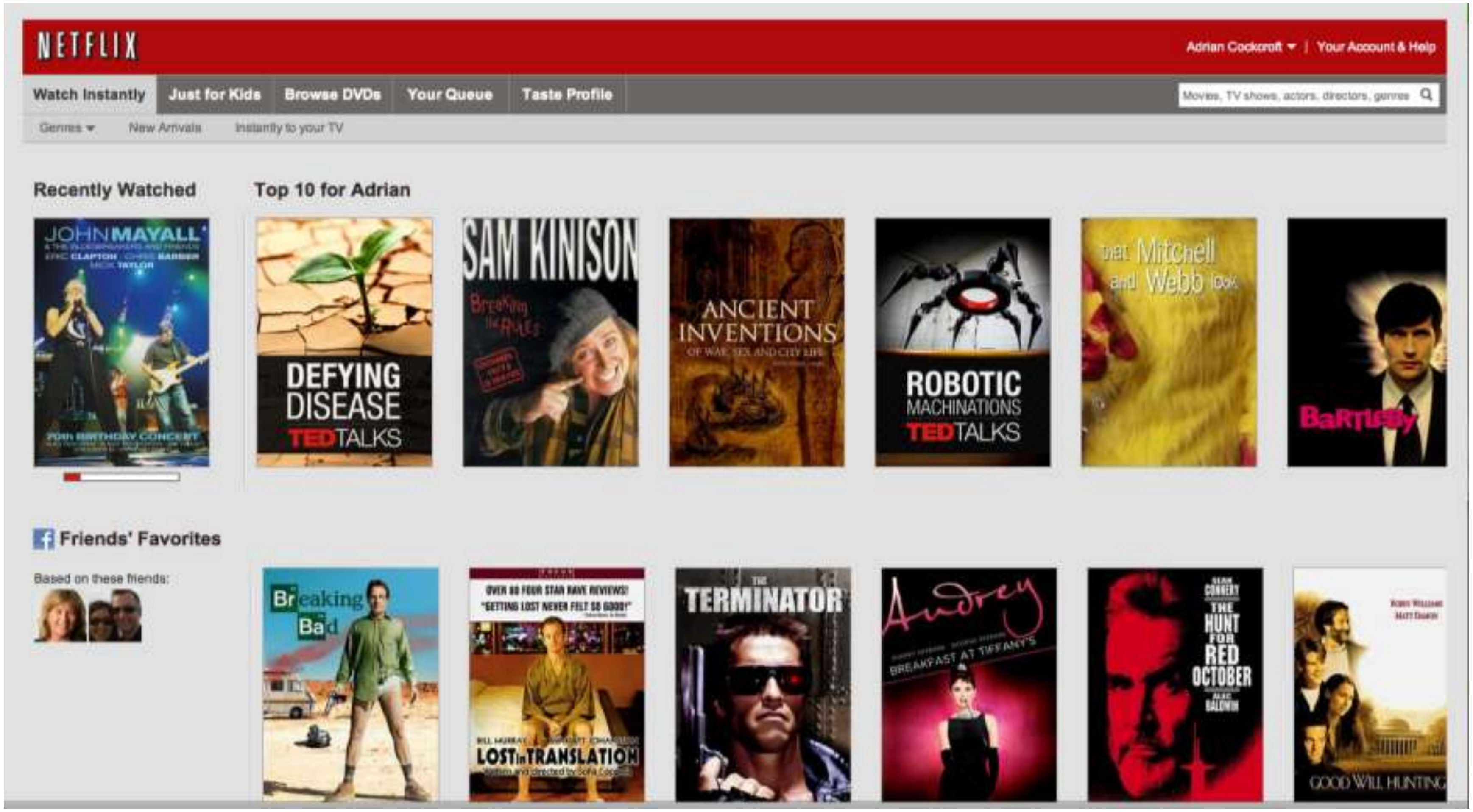
Ephemeral Instances

- Largest services are autoscaled
- Average lifetime of an instance is 36 hours

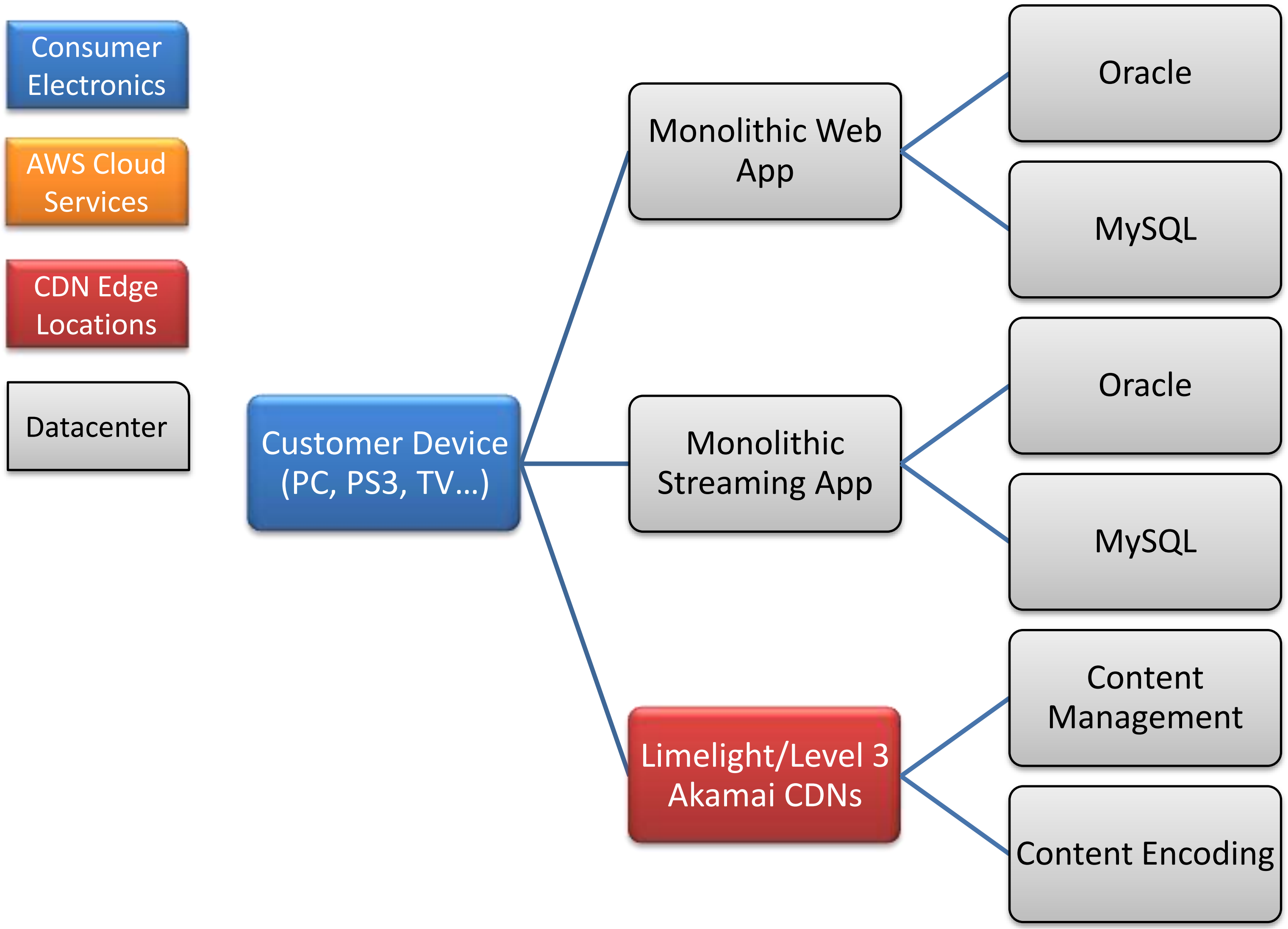


Netflix Member Web Site Home Page

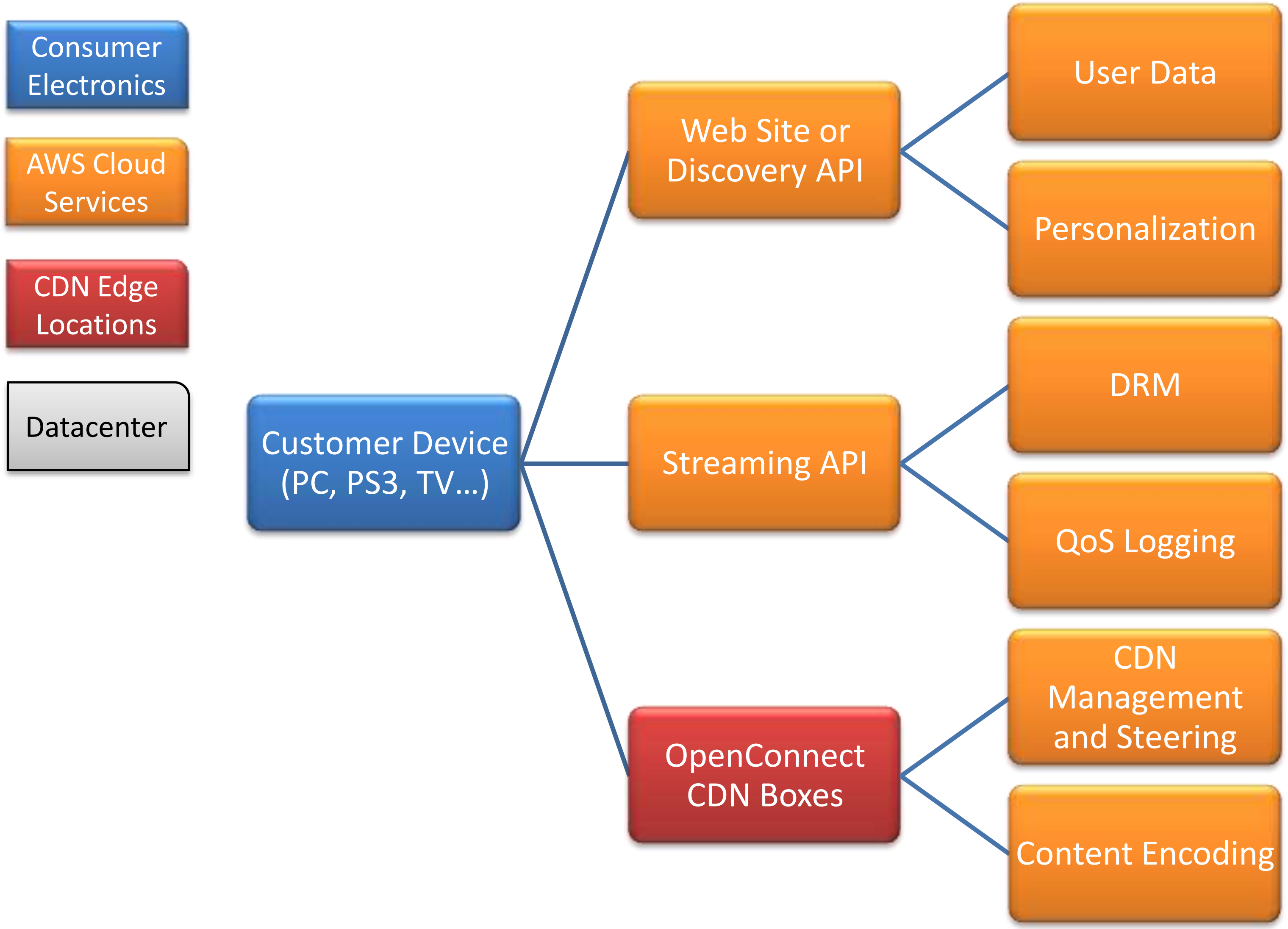
Personalization Driven – How Does It Work?



How Netflix Used to Work



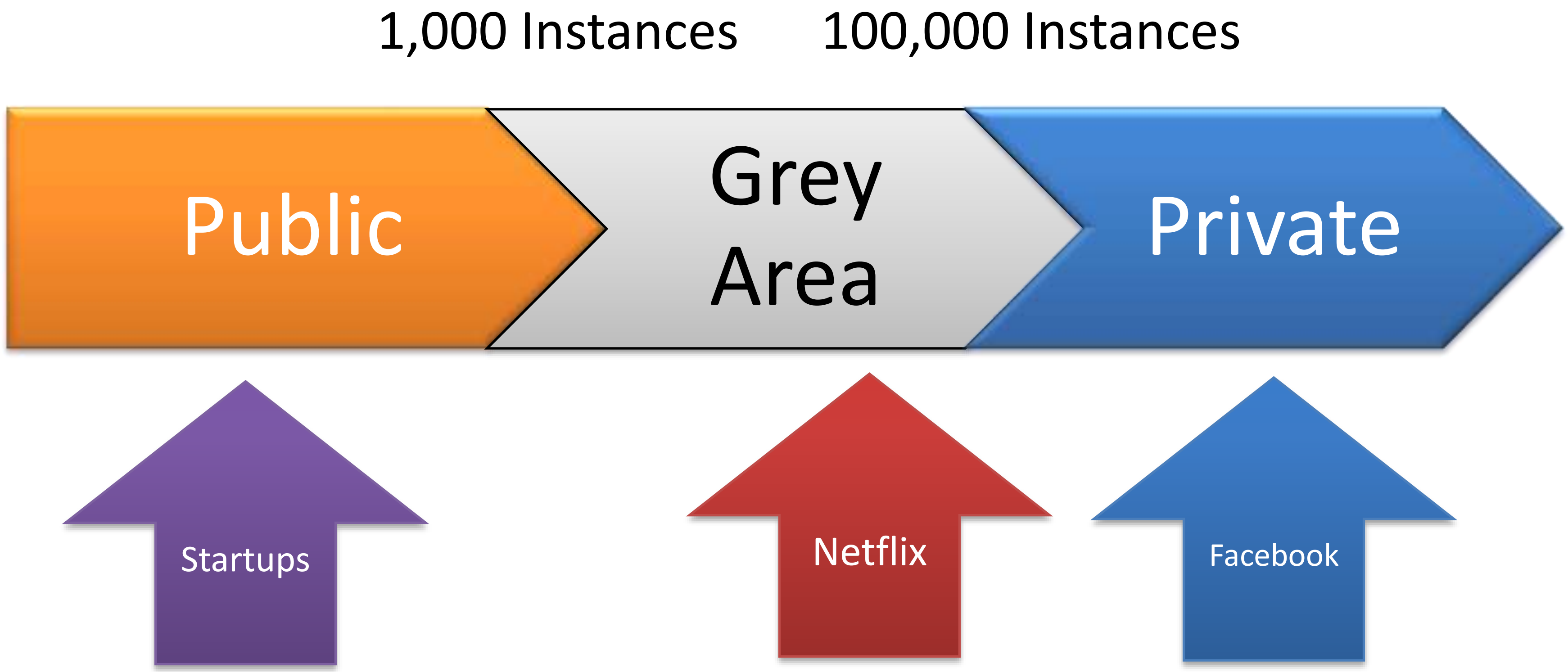
How Netflix Streaming Works Today



The DIY Question

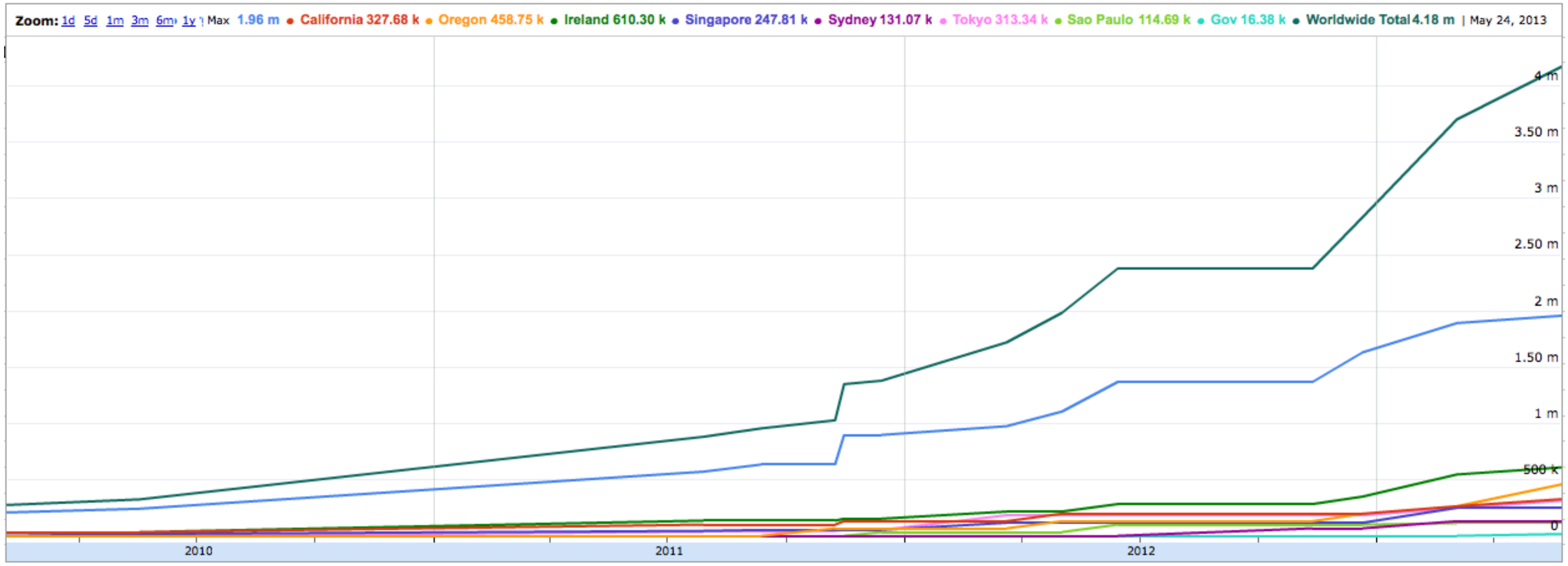
Why doesn’t Netflix build and run its own cloud?

Fitting Into Public Scale



How big is Public?

AWS Maximum Possible Instance Count 4.2 Million – May 2013
Growth >10x in Three Years, >2x Per Annum - http://bit.ly/awsiprange



AWS upper bound estimate based on the number of public IP Addresses
Every provisioned instance gets a public IP by default (some VPC don’t)

The Alternative Supplier Question

What if there is no clear leader for a feature, or AWS doesn’t have what we need?

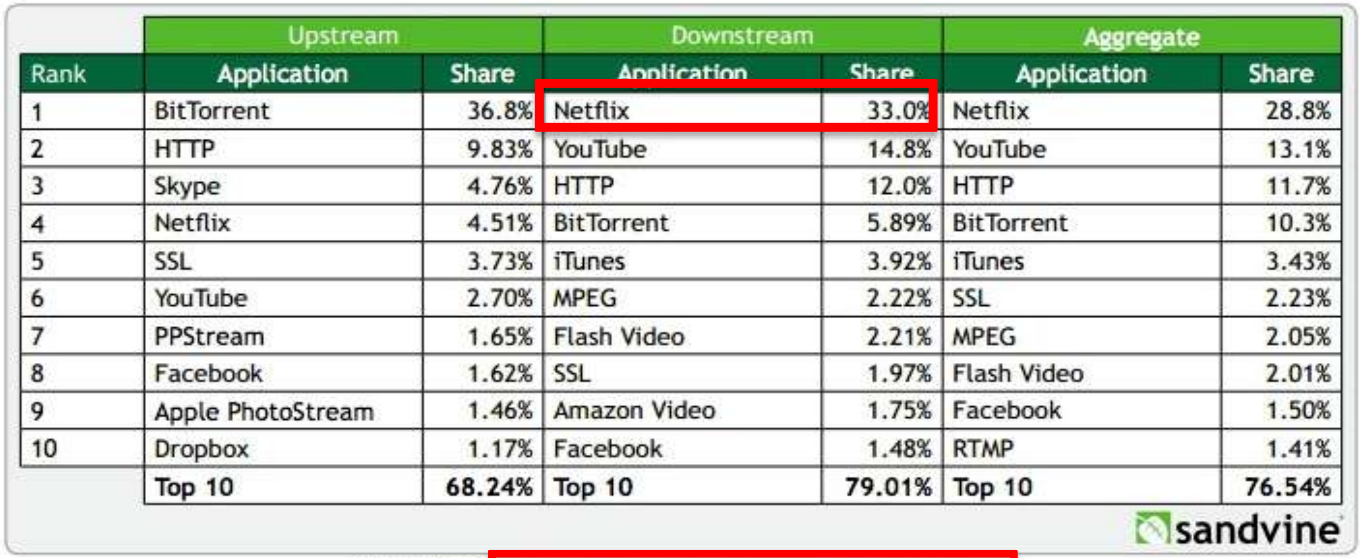
Things We Don’t Use AWS For

SaaS Applications – Pagerduty, Appdynamics

Content Delivery Service

DNS Service

Nov 2012 Streaming Bandwidth



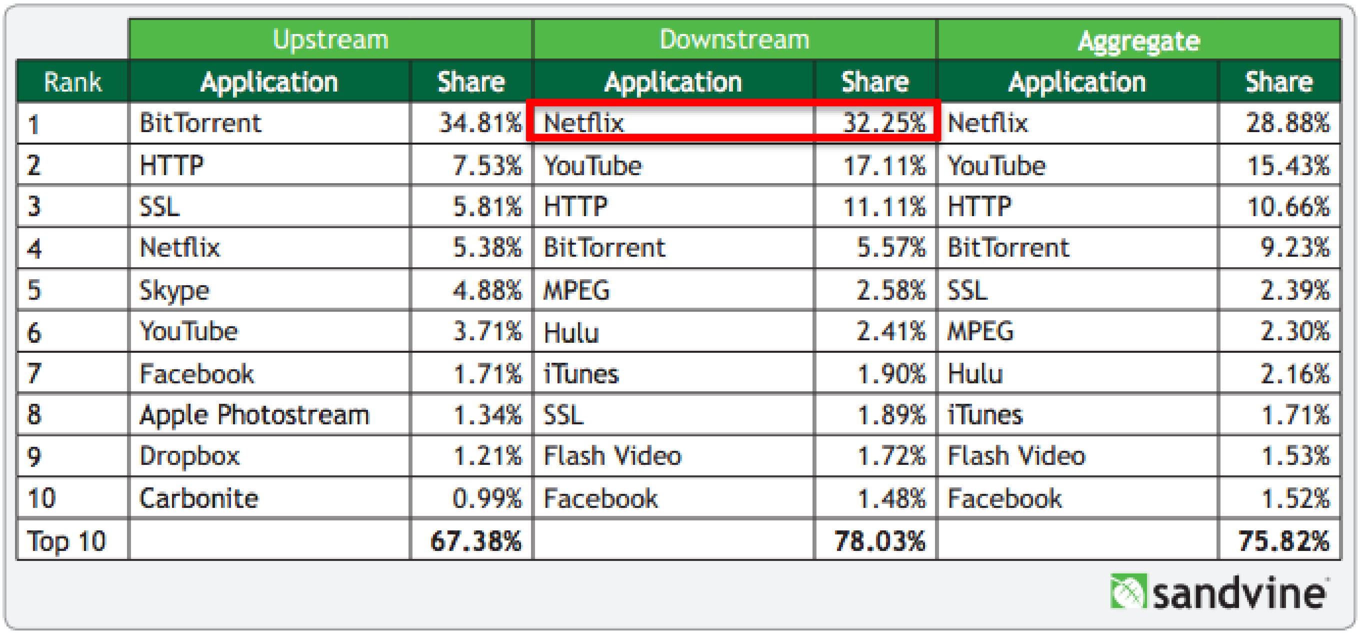
	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	36.8%	Netflix	33.0%	Netflix	28.8%
2	HTTP	9.83%	YouTube	14.8%	YouTube	13.1%
3	Skype	4.76%	HTTP	12.0%	HTTP	11.7%
4	Netflix	4.51%	BitTorrent	5.89%	BitTorrent	10.3%
5	SSL	3.73%	iTunes	3.92%	iTunes	3.43%
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

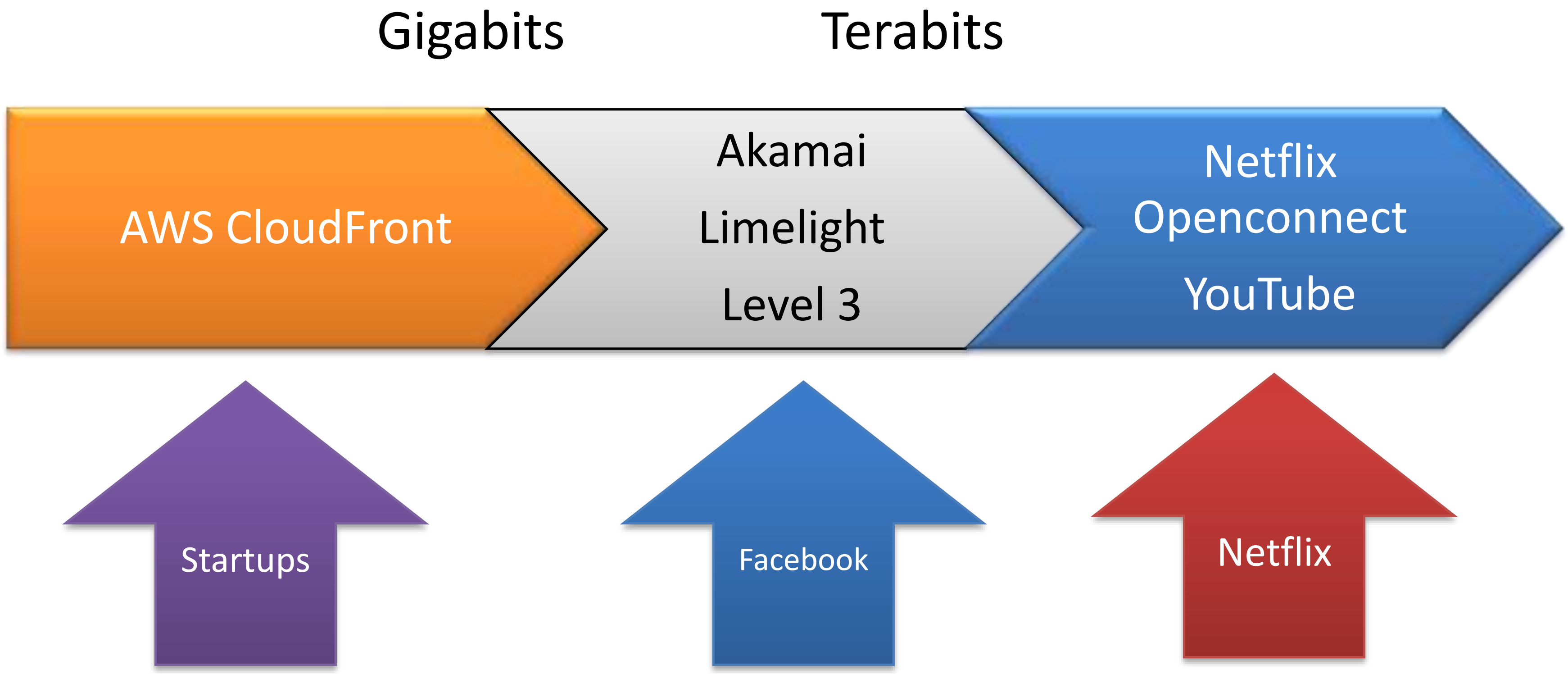
Mean Bandwidth +39% 6mo



	Upstream		Downstream		Aggregate	
Rank	Application	Share	Application	Share	Application	Share
1	BitTorrent	34.81%	Netflix	32.25%	Netflix	28.88%
2	HTTP	7.53%	YouTube	17.11%	YouTube	15.43%
3	SSL	5.81%	HTTP	11.11%	HTTP	10.66%
4	Netflix	5.38%	BitTorrent	5.57%	BitTorrent	9.23%
5	Skype	4.88%	MPEG	2.58%	SSL	2.39%
6	YouTube	3.71%	Hulu	2.41%	MPEG	2.30%
7	Facebook	1.71%	iTunes	1.90%	Hulu	2.16%
8	Apple Photostream	1.34%	SSL	1.89%	iTunes	1.71%
9	Dropbox	1.21%	Flash Video	1.72%	Flash Video	1.53%
10	Carbonite	0.99%	Facebook	1.48%	Facebook	1.52%
Top 10		67.38%		78.03%		75.82%

sandvine

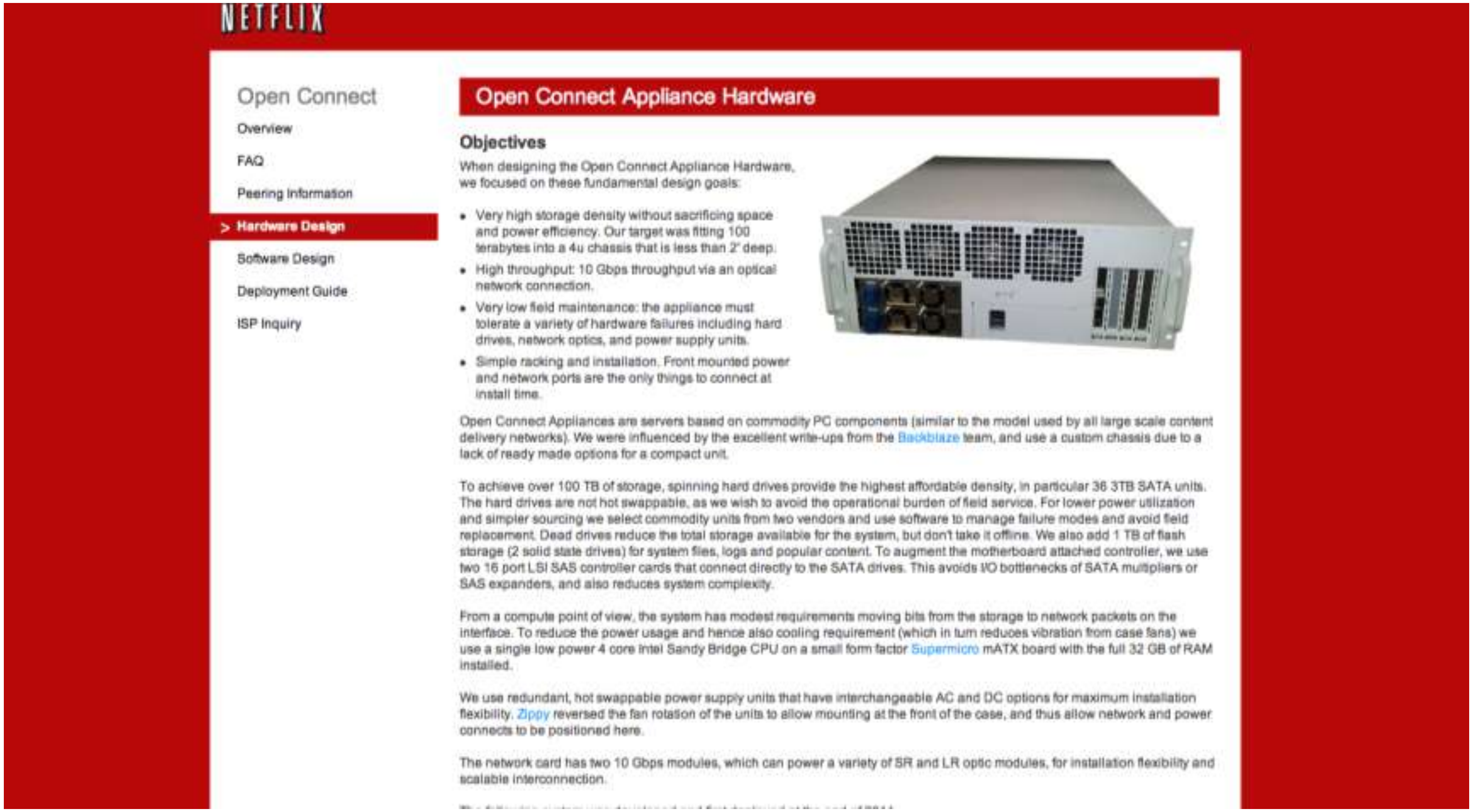
CDN Scale


Gigabits
Terabits
AWS CloudFront
Akamai
Limelight
Level 3
Netflix
Openconnect
YouTube
Startups
Facebook
Netflix

Content Delivery Service

Open Source Hardware Design + FreeBSD, bird, nginx

see openconnect.netflix.com



Open Connect Appliance Hardware

Objectives

When designing the Open Connect Appliance Hardware, we focused on these fundamental design goals:

- Very high storage density without sacrificing space and power efficiency. Our target was fitting 100 terabytes into a 4u chassis that is less than 2' deep.
- High throughput: 10 Gbps throughput via an optical network connection.
- Very low field maintenance: the appliance must tolerate a variety of hardware failures including hard drives, network optics, and power supply units.
- Simple racking and installation. Front mounted power and network ports are the only things to connect at install time.

Open Connect Appliances are servers based on commodity PC components (similar to the model used by all large scale content delivery networks). We were influenced by the excellent write-ups from the Backblaze team, and use a custom chassis due to a lack of ready made options for a compact unit.

To achieve over 100 TB of storage, spinning hard drives provide the highest affordable density, in particular 36 3TB SATA units. The hard drives are not hot swappable, as we wish to avoid the operational burden of field service. For lower power utilization and simpler sourcing we select commodity units from two vendors and use software to manage failure modes and avoid field replacement. Dead drives reduce the total storage available for the system, but don't take it offline. We also add 1 TB of flash storage (2 solid state drives) for system files, logs and popular content. To augment the motherboard attached controller, we use two 16 port LSI SAS controller cards that connect directly to the SATA drives. This avoids I/O bottlenecks of SATA multipliers or SAS expanders, and also reduces system complexity.

From a compute point of view, the system has modest requirements moving bits from the storage to network packets on the interface. To reduce the power usage and hence also cooling requirement (which in turn reduces vibration from case fans) we use a single low power 4 core Intel Sandy Bridge CPU on a small form factor Supermicro mATX board with the full 32 GB of RAM installed.

We use redundant, hot swappable power supply units that have interchangeable AC and DC options for maximum installation flexibility. Zippy reversed the fan rotation of the units to allow mounting at the front of the case, and thus allow network and power connects to be positioned here.

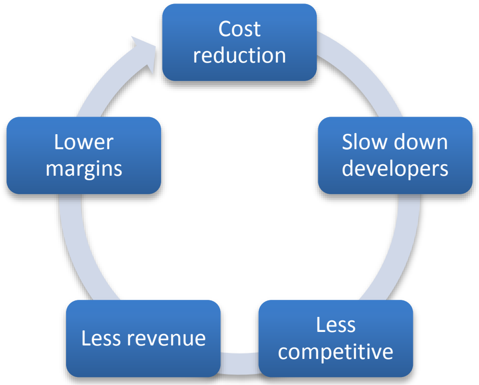
The network card has two 10 Gbps modules, which can power a variety of SR and LR optic modules, for installation flexibility and scalable interconnection.

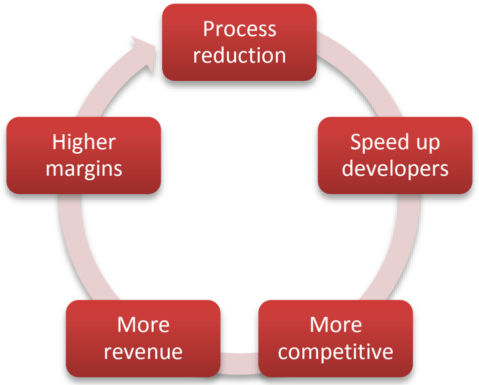
DNS Service

AWS Route53 is missing too many features (for now)

Multiple vendor strategy Dyn, Ultra, Route53

Abstracted (broken) DNS APIs with Denominator





What Changed?

Get out of the way of innovation

Best of breed, by the hour

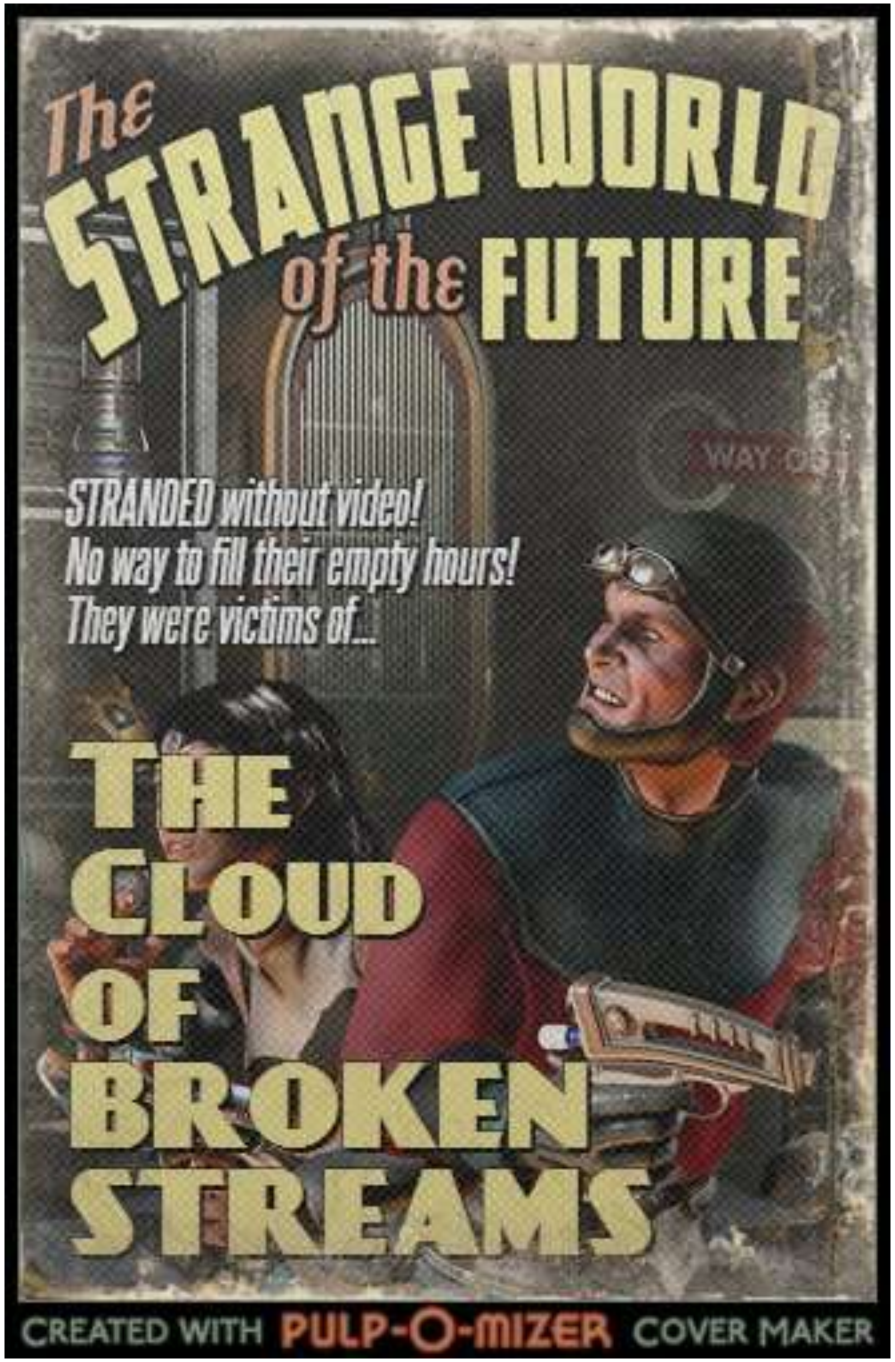
Choices based on scale

Availability Questions

Is it running yet?

How many places is it running in?

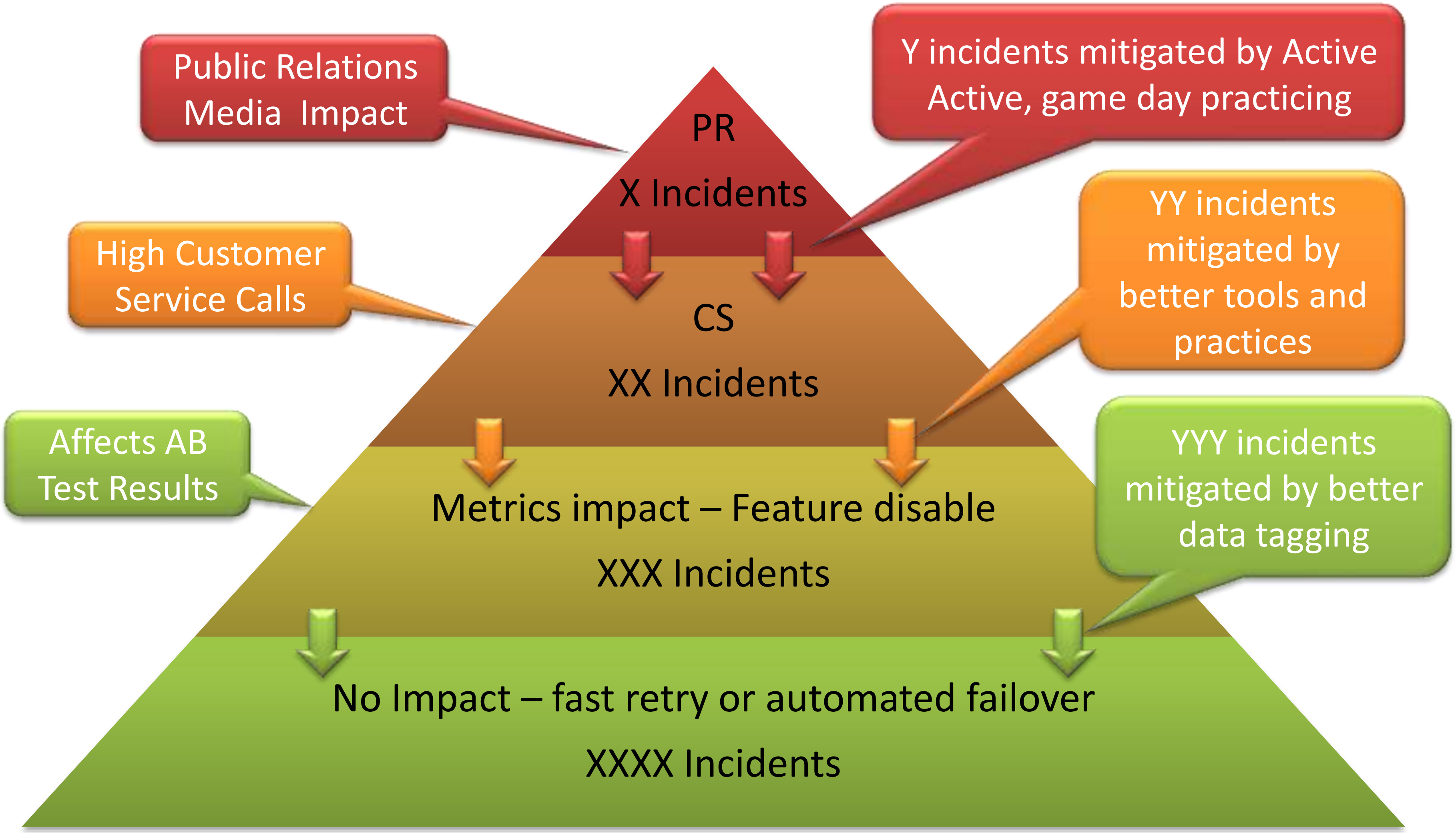
How far apart are those places?


The STRANGE WORLD of the FUTURE
WAY OUT
STRANDED without video!
No way to fill their empty hours!
They were victims of...
THE CLOUD OF BROKEN STREAMS
CREATED WITH PULP-O-MIZER COVER MAKER

Netflix Outages

- Running very fast with scissors
  - Mostly self inflicted – bugs, mistakes from pace of change
  - Some caused by AWS bugs and mistakes
- Incident Life-cycle Management by Platform Team
  - No runbooks, no operational changes by the SREs
  - Tools to identify what broke and call the right developer
- Next step is multi-region active/active
  - Investigating and building in stages during 2013
  - Could have prevented some of our 2012 outages

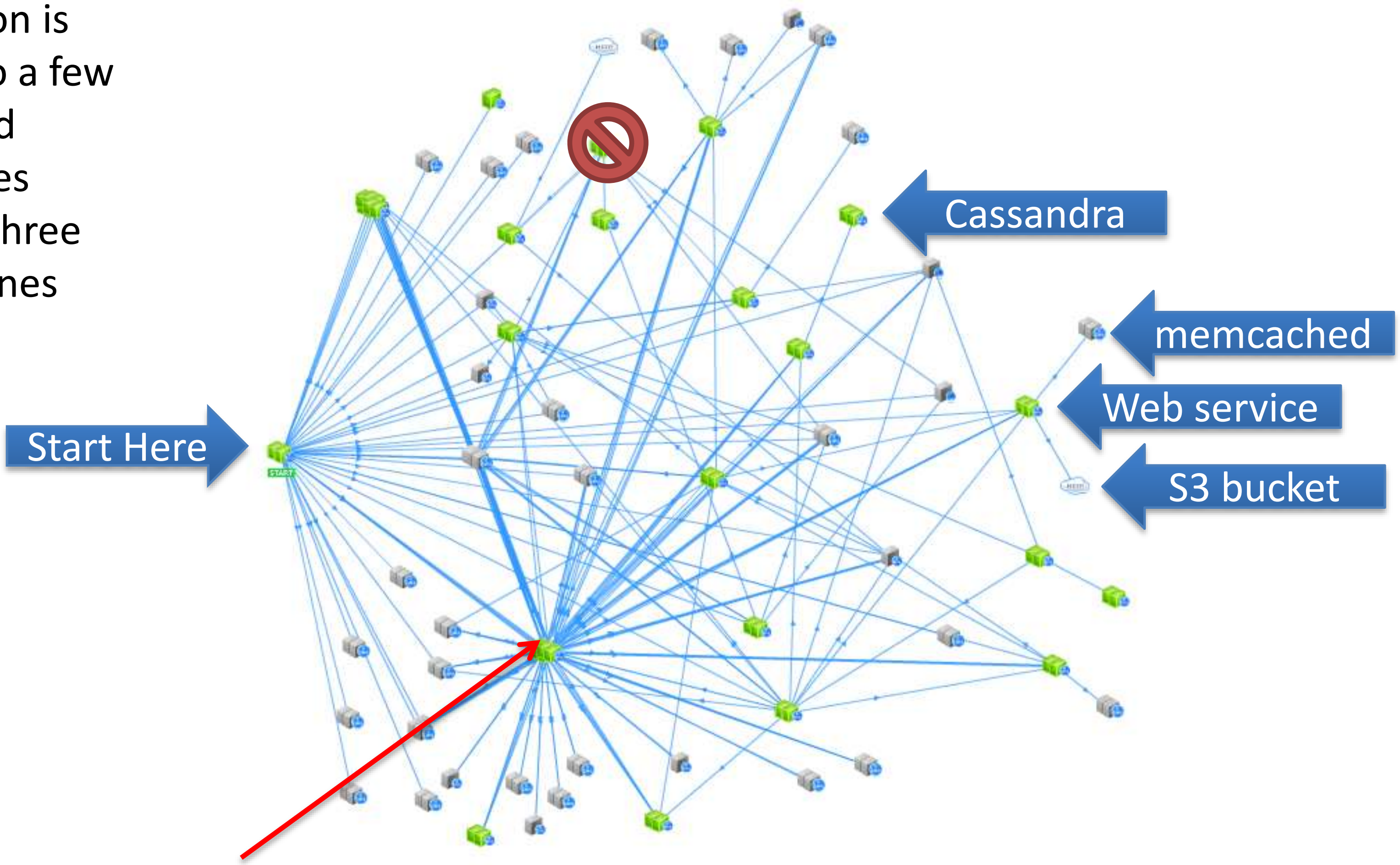
Incidents – Impact and Mitigation


Public Relations Media Impact
PR
X Incidents
Y incidents mitigated by Active Active, game day practicing
High Customer Service Calls
CS
XX Incidents
YY incidents mitigated by better tools and practices
Affects AB Test Results
Metrics impact – Feature disable
XXX Incidents
YYY incidents mitigated by better data tagging
No Impact – fast retry or automated failover
XXXX Incidents

Real Web Server Dependencies Flow

(Netflix Home page business transaction as seen by AppDynamics)

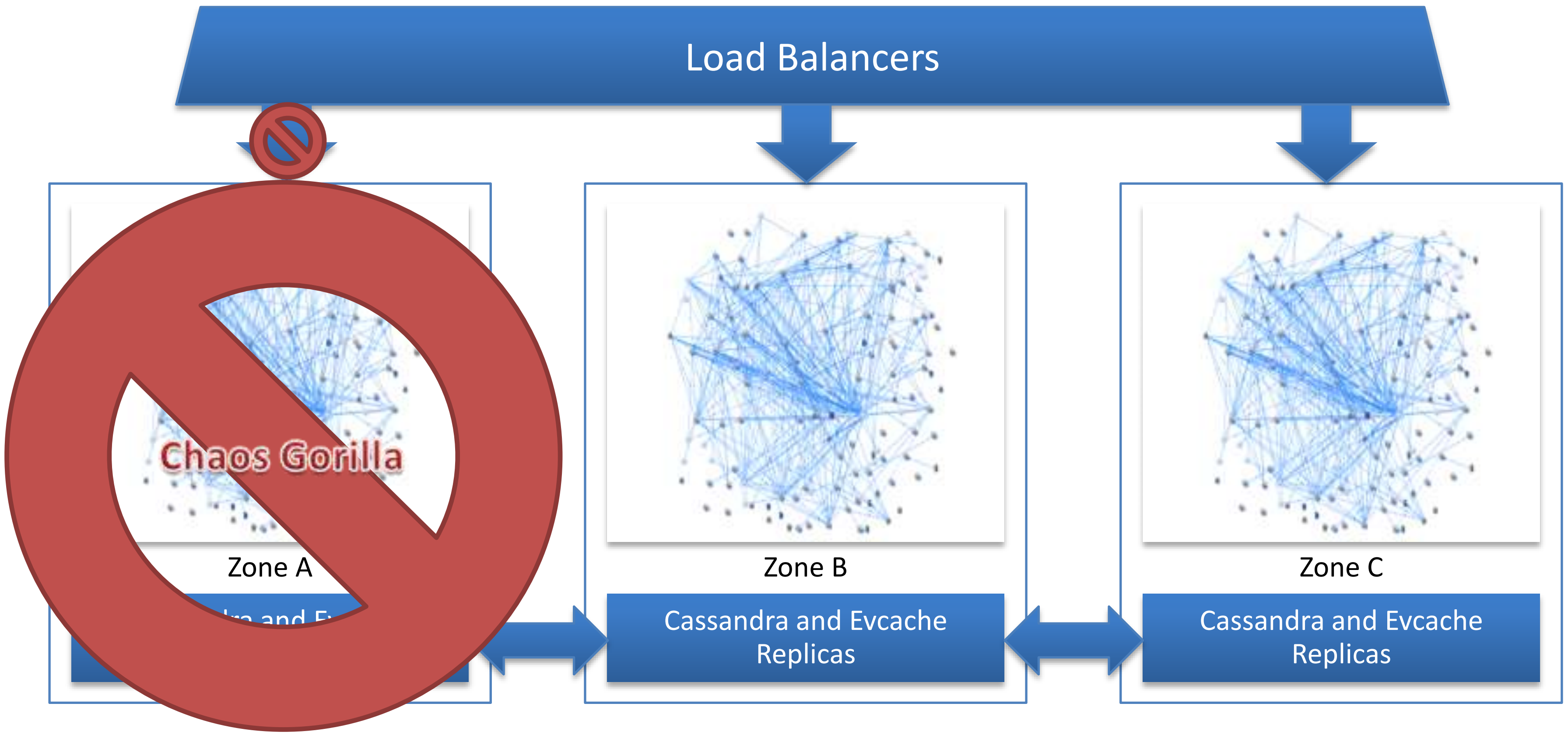
Each icon is three to a few hundred instances across three AWS zones



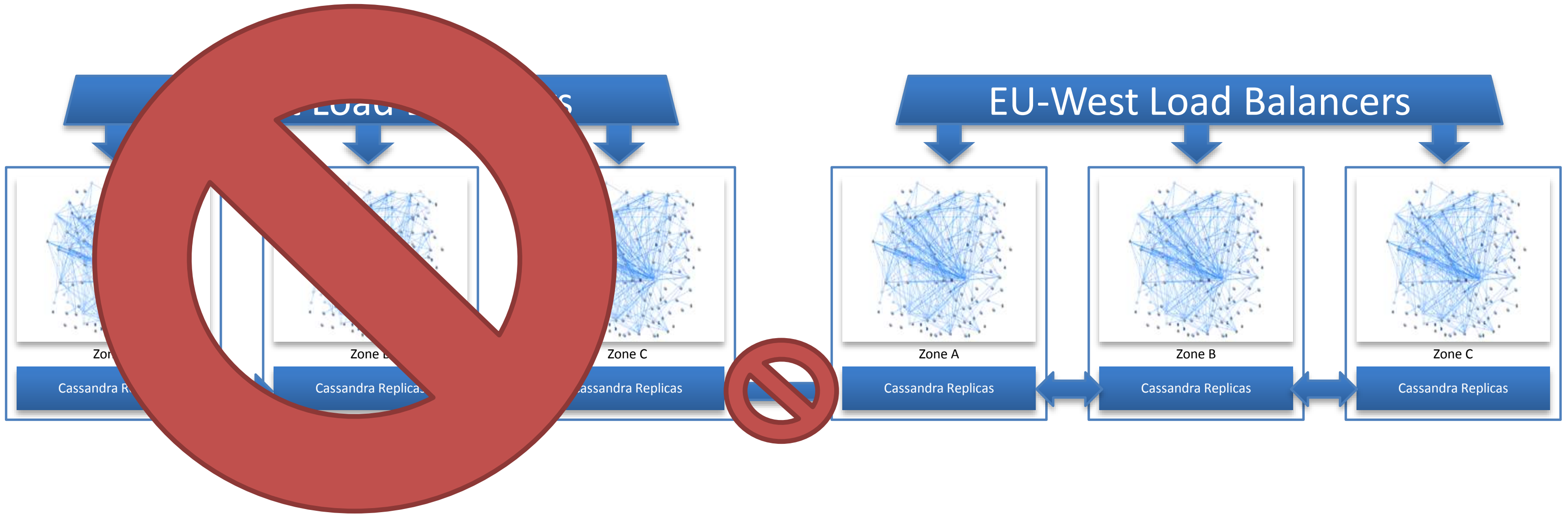
Personalization movie group choosers (for US, Canada and Latam)

Three Balanced Availability Zones

Test with Chaos Gorilla


Load Balancers
Chaos Gorilla
Zone A
Zone B
Zone C
Cassandra and Evcache Replicas
Cassandra and Evcache Replicas

Isolated Regions



More?

Highly Available NoSQL Storage

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

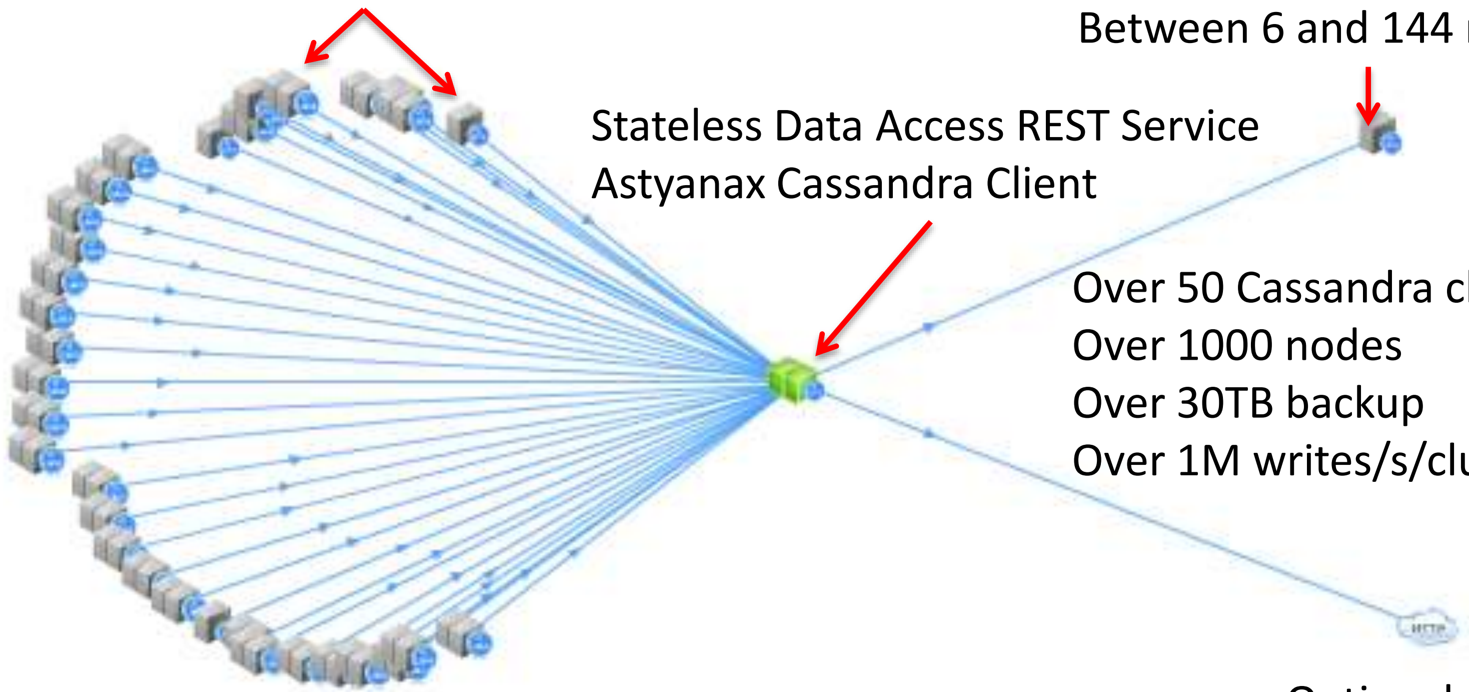
Single Function Micro-Service Pattern

One keyspace, replaces a single table or materialized view

Many Different Single-Function REST Clients

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

Stateless Data Access REST Service
Astyanax Cassandra Client



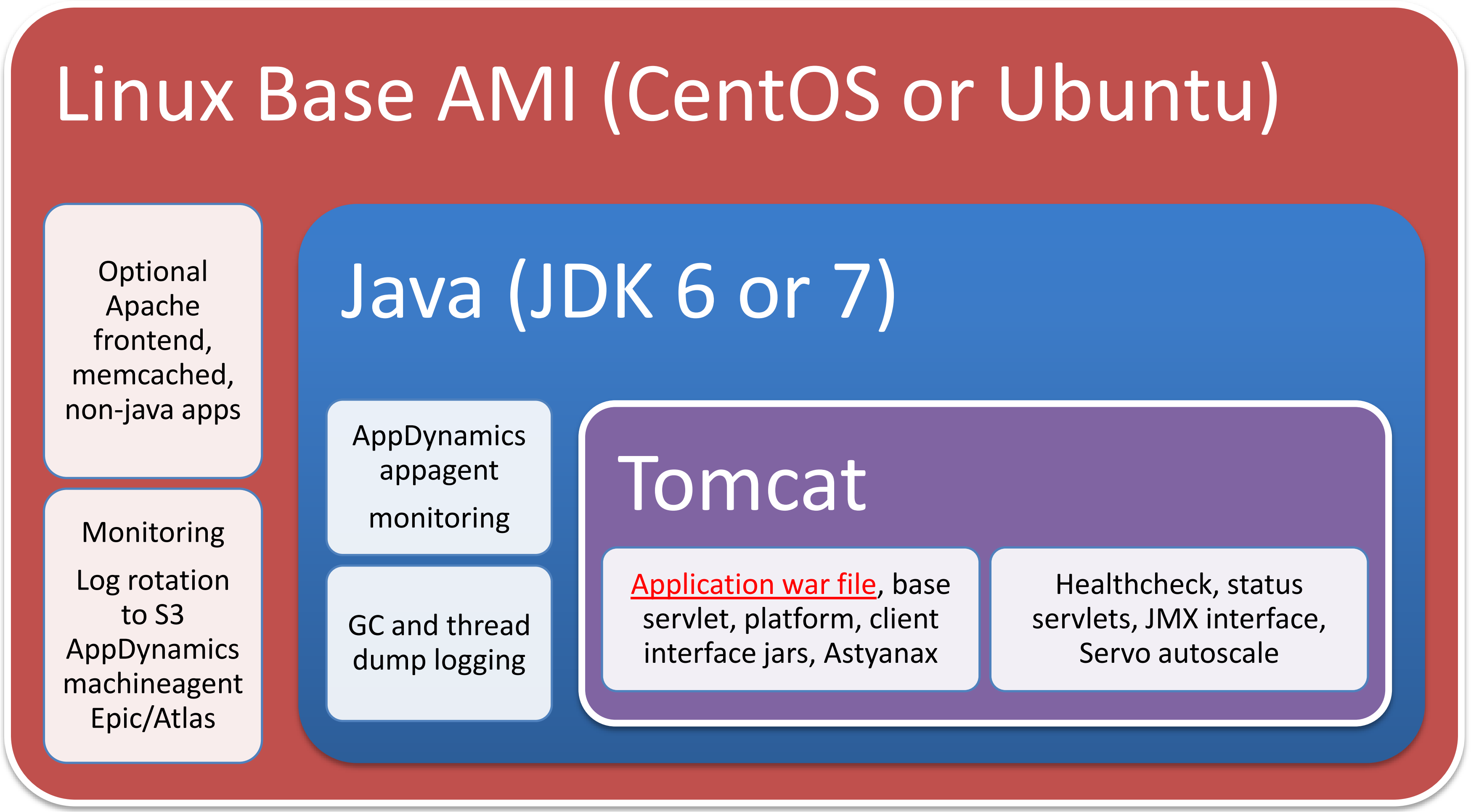
Over 50 Cassandra clusters
Over 1000 nodes
Over 30TB backup
Over 1M writes/s/cluster

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

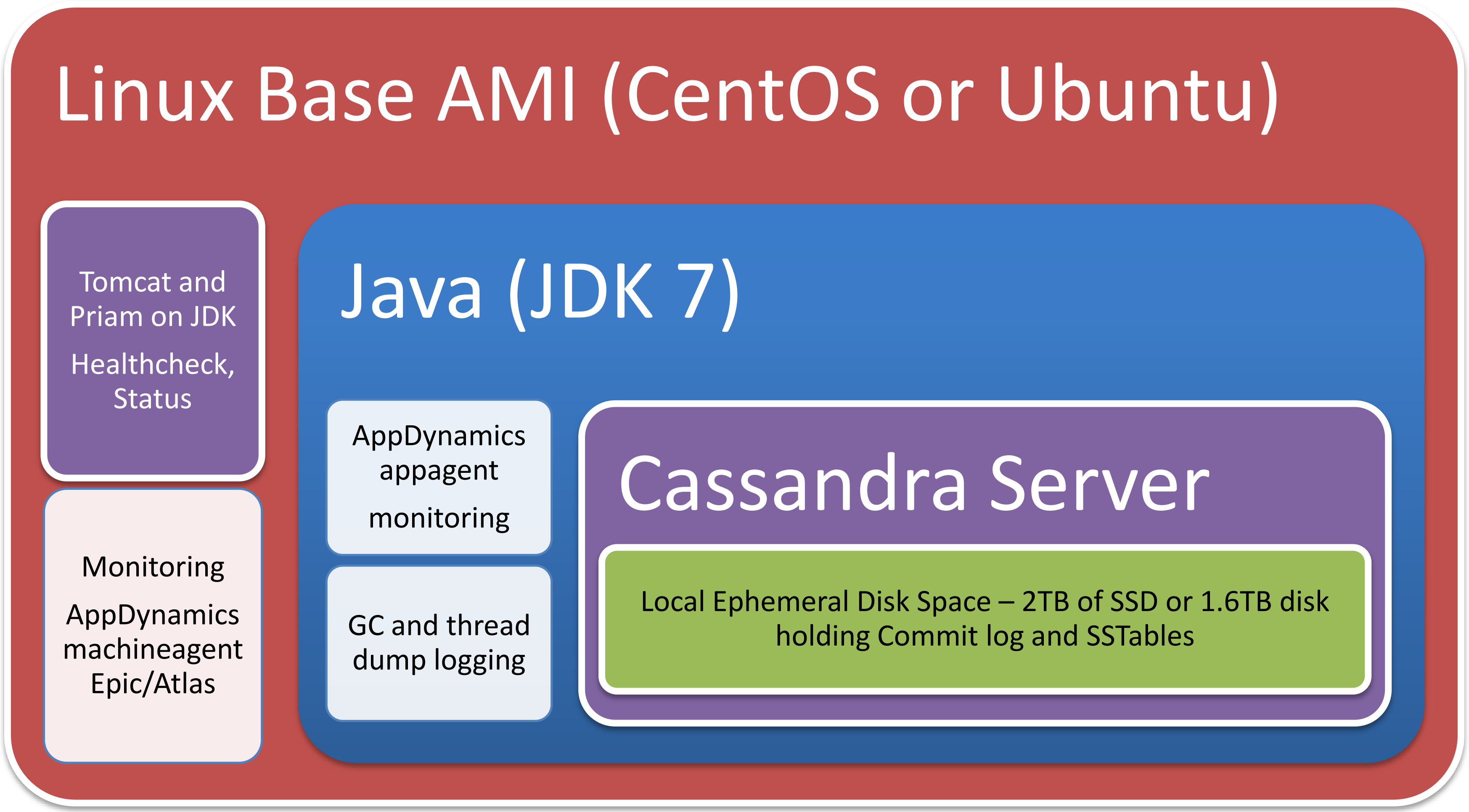
Optional
Datacenter
Update Flow

Appdynamics Service Flow Visualization

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

Apache Cassandra

- Scalable and Stable in large deployments
  - No additional license cost for large scale!
  - Optimized for “OLTP” vs. Hbase optimized for “DSS”
- Available during Partition (AP from CAP)
  - Hinted handoff repairs most transient issues
  - Read-repair and periodic repair keep it clean
- Quorum and Client Generated Timestamp
  - Read after write consistency with 2 of 3 copies
  - Latest version includes Paxos for stronger transactions

Astyanax Cassandra Client for Java

Available at http://github.com/netflix

- Features
  - Abstraction of connection pool from RPC protocol
  - Fluent Style API
  - Operation retry with backoff
  - Token aware
  - Batch manager
  - Many useful recipes
  - New: Entity Mapper based on JPA annotations

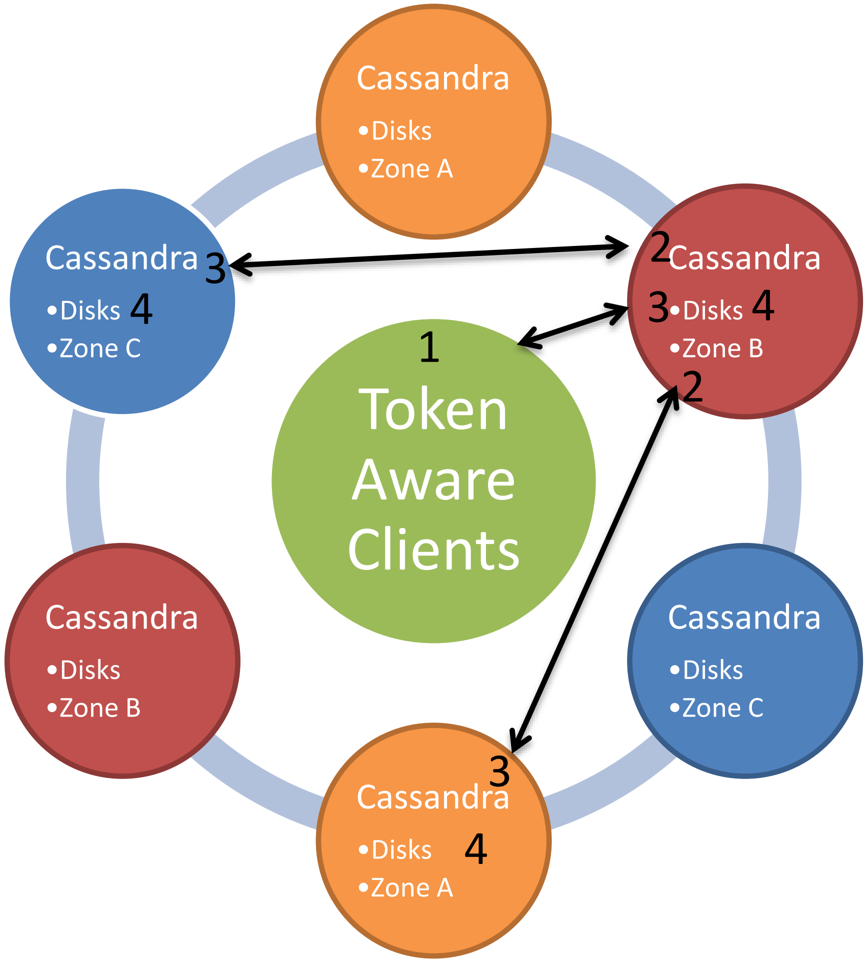
C* Astyanax Recipes

- Distributed row lock (without needing 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
- Contributed: High cardinality reverse index

Astyanax - Cassandra Write Data Flows

Single Region, Multiple Availability Zone, Token Aware

1. Client Writes to local coordinator
2. Coodinator 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

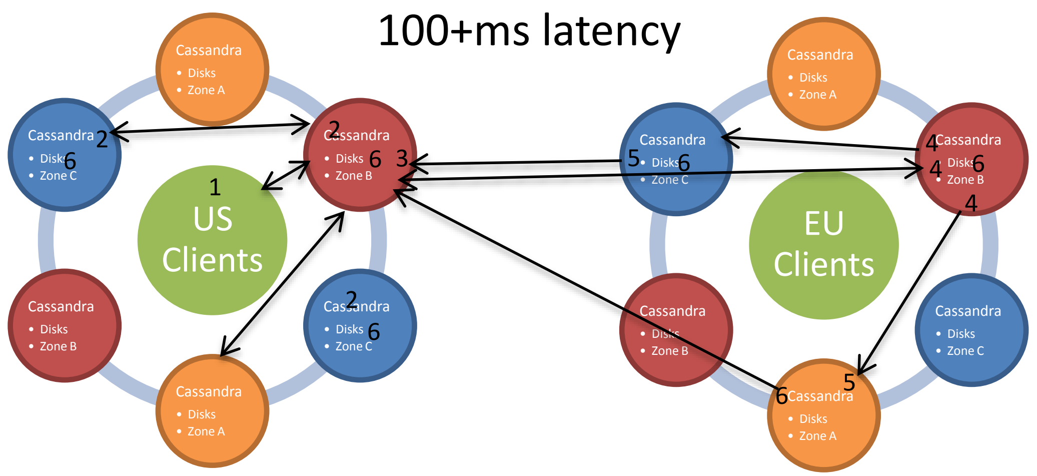
SSTable disk writes and compactions occur asynchronously

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.



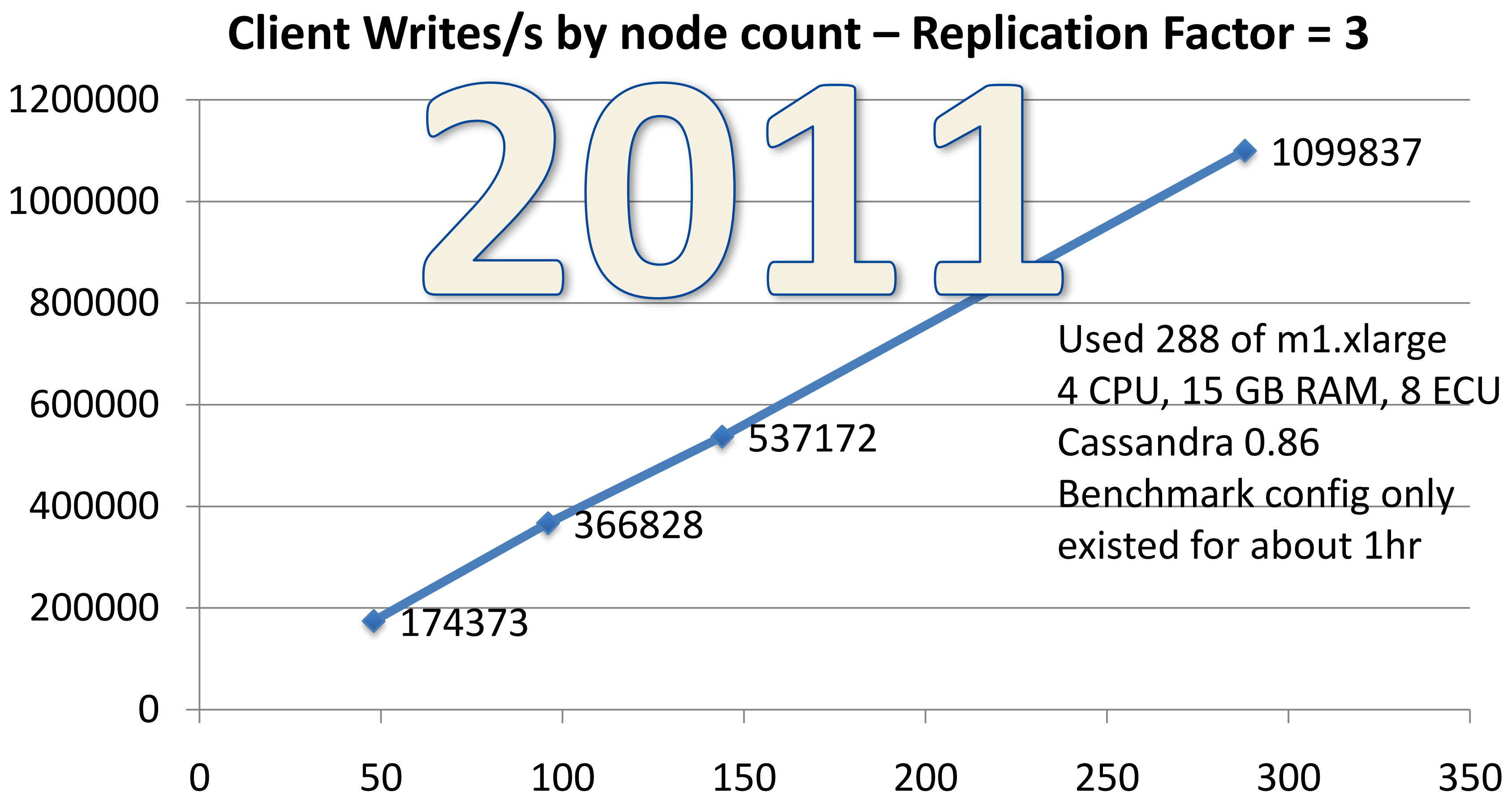
Cassandra at Scale

Benchmarking to Retire Risk

More?

Scalability from 48 to 288 nodes on AWS

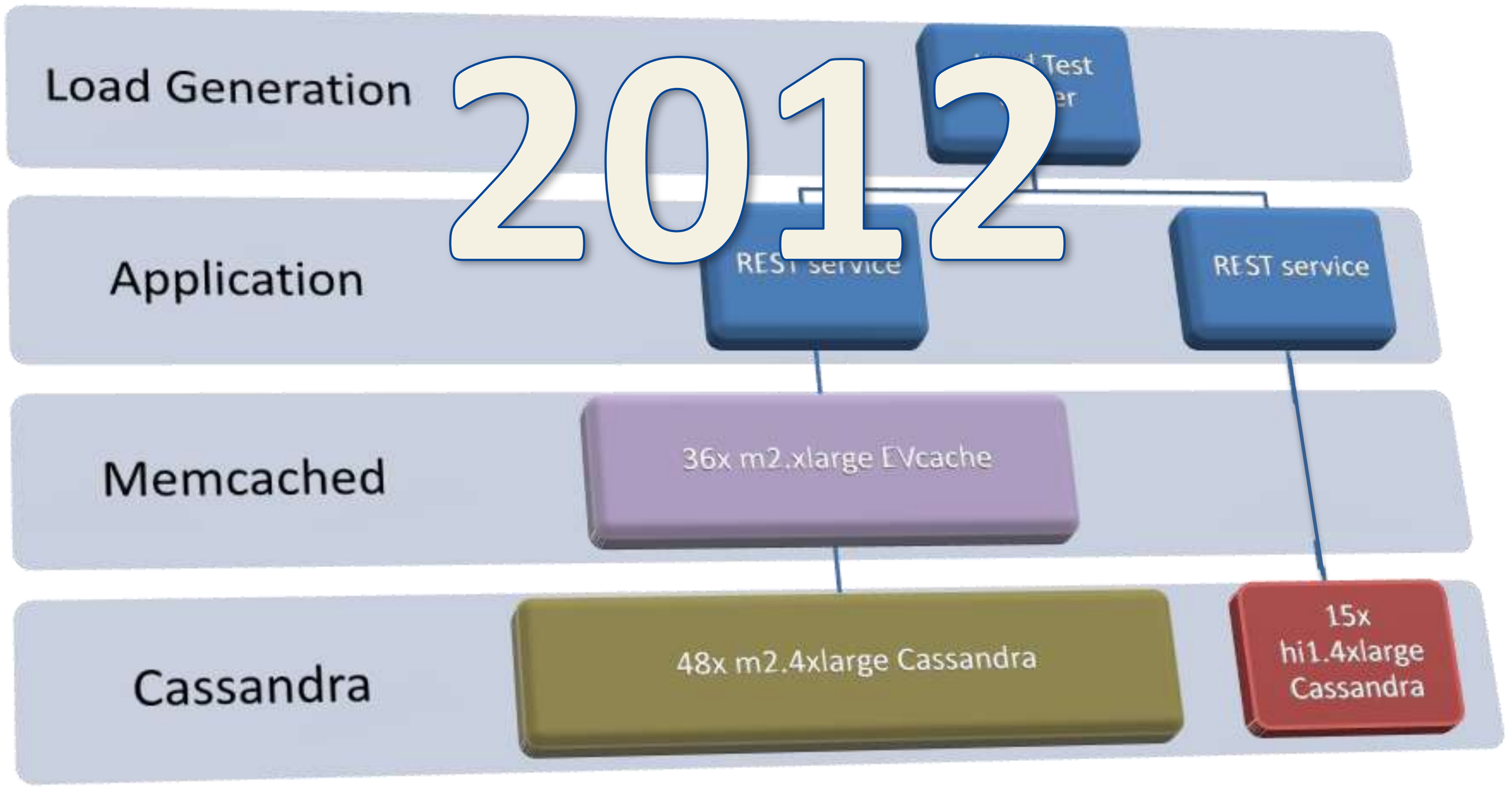
http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html



Cassandra Disk vs. SSD Benchmark

Same Throughput, Lower Latency, Half Cost

http://techblog.netflix.com/2012/07/benchmarking-high-performance-io-with.html

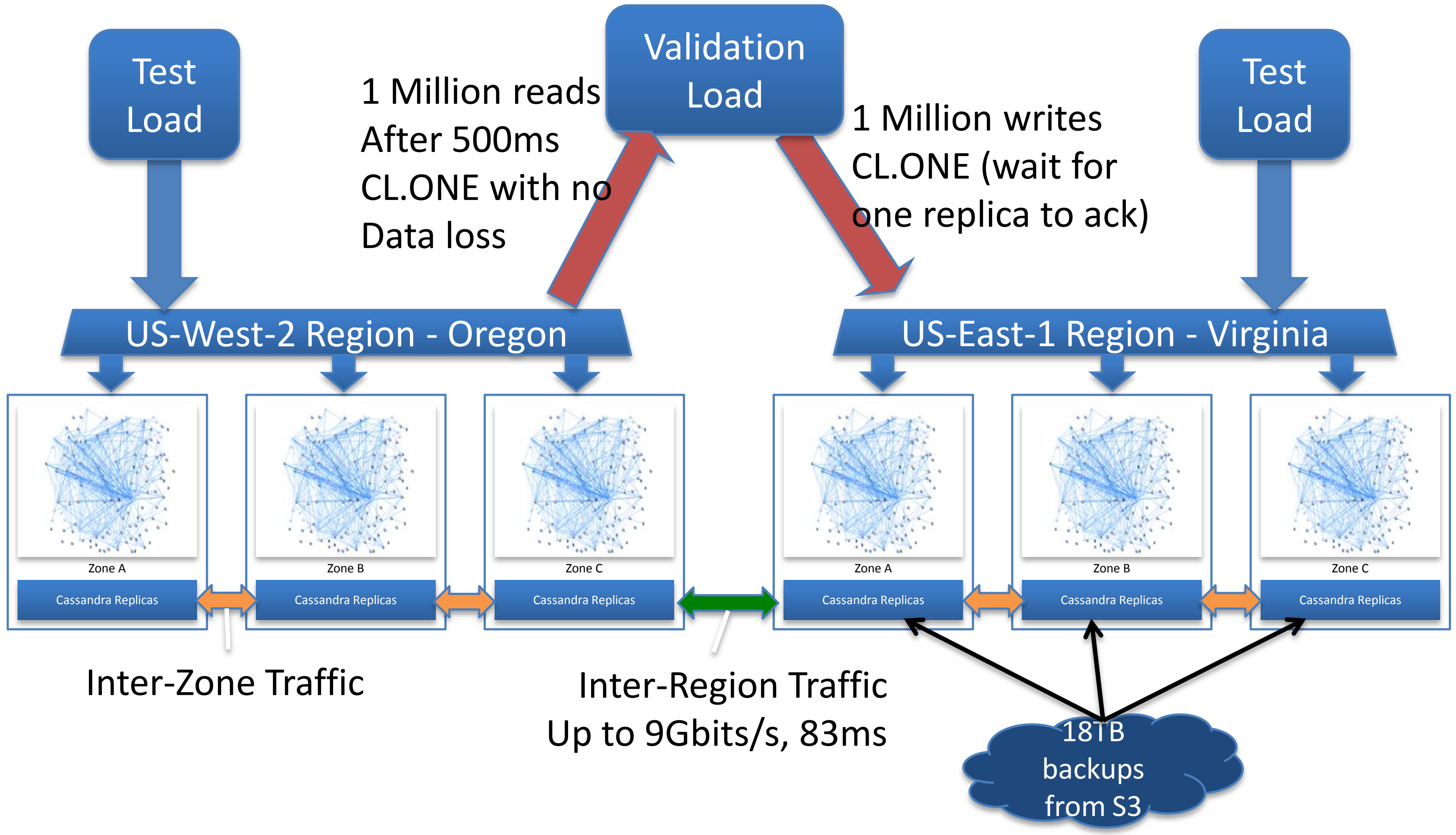


2013 - Cross Region Use Cases

- 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

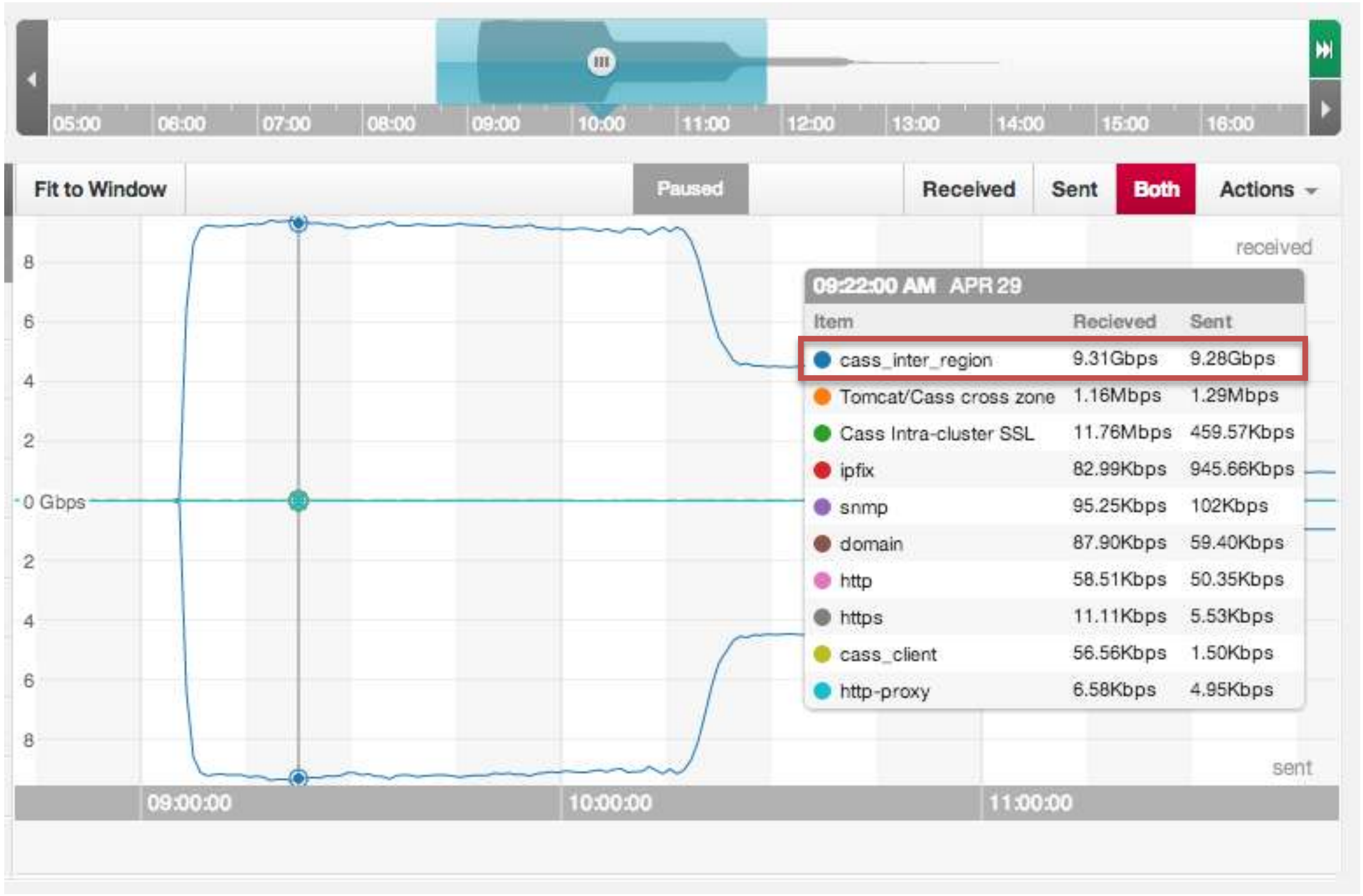
Benchmarking Global Cassandra

Write intensive test of cross region replication capacity
16 x hi1.4xlarge SSD nodes per zone = 96 total
192 TB of SSD in six locations up and running Cassandra <u>in 20 minutes</u>


Test Load
Validation Load
Test Load
1 Million reads After 500ms CL.ONE with no Data loss
1 Million writes CL.ONE (wait for one replica to ack)
US-West-2 Region - Oregon
US-East-1 Region - Virginia
Zone A
Zone B
Zone C
Cassandra Replicas
Inter-Zone Traffic
Inter-Region Traffic Up to 9Gbits/s, 83ms
18TB backups from S3

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

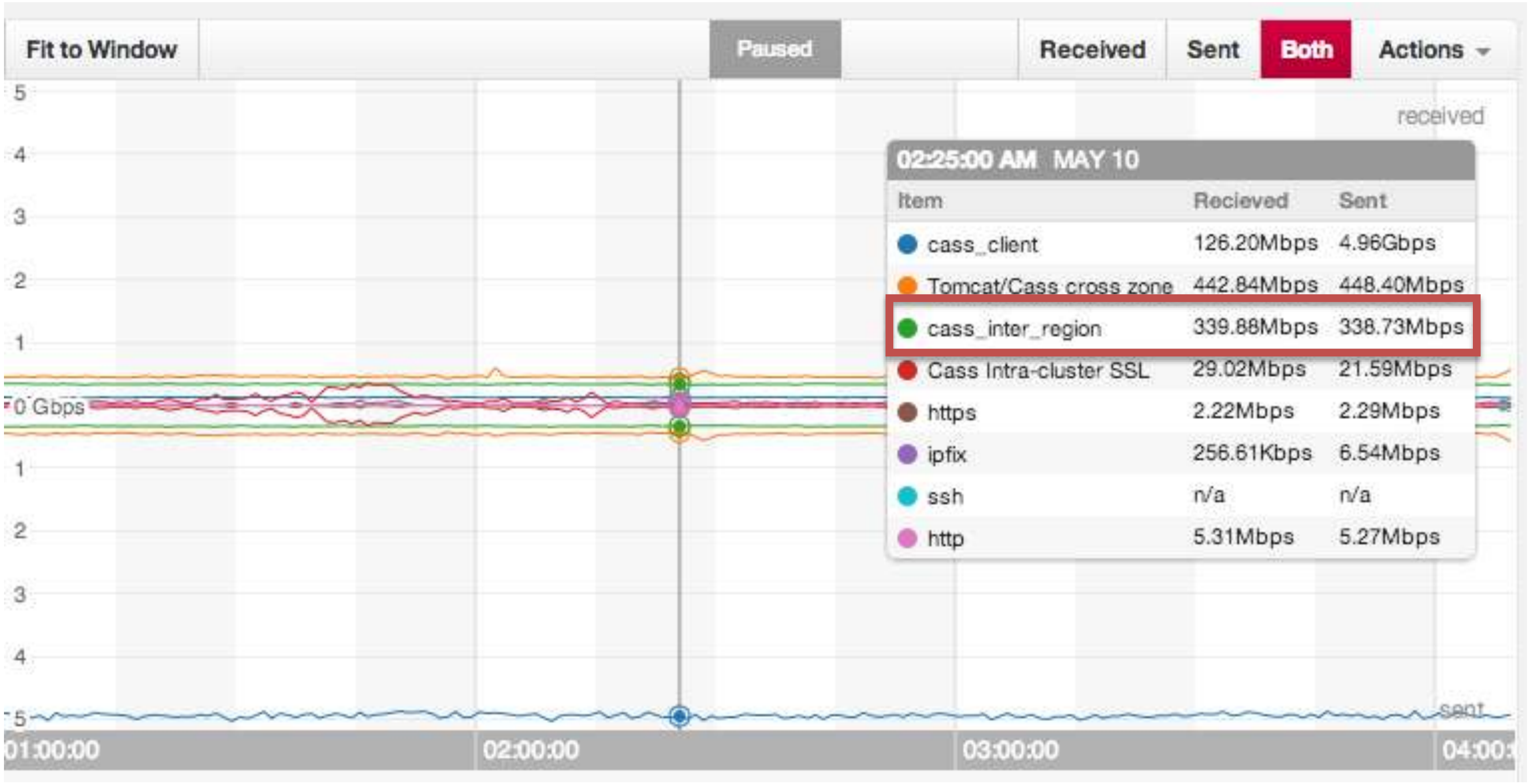


09:22:00 AM APR 29

Item	Recieved	Sent
● cass_inter_region	9.31Gbps	9.28Gbps
● Tomcat/Cass cross zone	1.16Mbps	1.29Mbps
● Cass Intra-cluster SSL	11.76Mbps	459.57Kbps
● ipfix	82.99Kbps	945.66Kbps
● snmp	95.25Kbps	102Kbps
● domain	87.90Kbps	59.40Kbps
● http	58.51Kbps	50.35Kbps
● https	11.11Kbps	5.53Kbps
● cass_client	56.56Kbps	1.50Kbps
● http-proxy	6.58Kbps	4.95Kbps

Inter Region Traffic Test

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

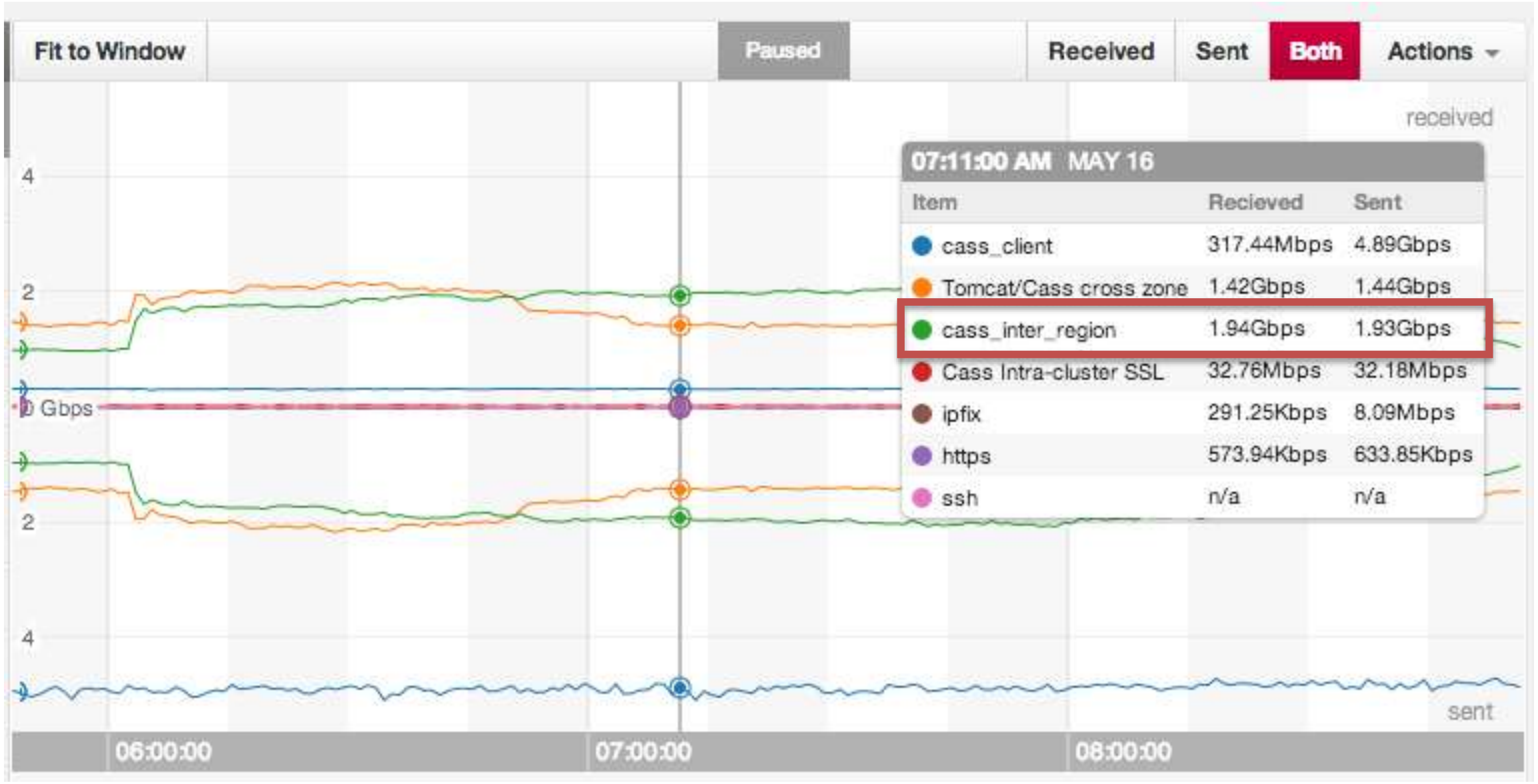


02:25:00 AM MAY 10

Item	Recieved	Sent
● cass_client	126.20Mbps	4.96Gbps
● Tomcat/Cass cross zone	442.84Mbps	448.40Mbps
● cass_inter_region	339.88Mbps	338.73Mbps
● Cass Intra-cluster SSL	29.02Mbps	21.59Mbps
● https	2.22Mbps	2.29Mbps
● ipfix	256.61Kbps	6.54Mbps
● ssh	n/a	n/a
● http	5.31Mbps	5.27Mbps

Ramp Up Load Until It Breaks!

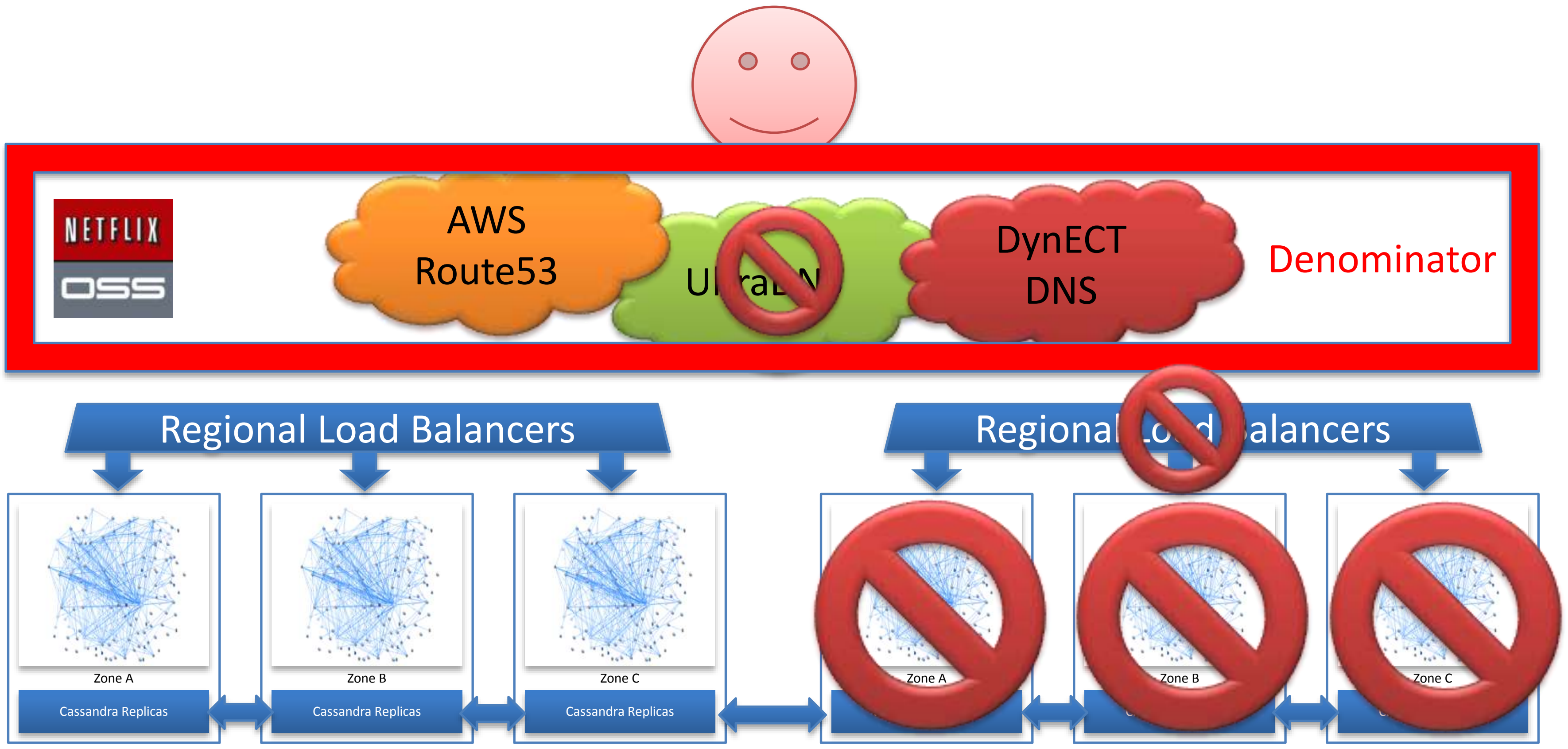
Unmodified tuning, dropping client data at 1.93GB/s inter region traffic
Spare CPU, IOPS, Network, just need some Cassandra tuning for more



07:11:00 AM MAY 16

Item	Recieved	Sent
● cass_client	317.44Mbps	4.89Gbps
● Tomcat/Cass cross zone	1.42Gbps	1.44Gbps
● cass_inter_region	1.94Gbps	1.93Gbps
● Cass Intra-cluster SSL	32.76Mbps	32.18Mbps
● ipfix	291.25Kbps	8.09Mbps
● https	573.94Kbps	633.85Kbps
● ssh	n/a	n/a

Managing Multi-Region Availability



Denominator – manage traffic via multiple DNS providers with Java code
2013 Timeline - Concept Jan, Code Feb, OSS March, Production use May

Failure Modes and Effects

Failure Mode	Probability	Current Mitigation Plan
Application Failure	High	Automatic degraded response
AWS Region Failure	Low	Active-Active multi-region deployment
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. Getting there…

Application Resilience

Run what you wrote

Rapid detection

Rapid Response

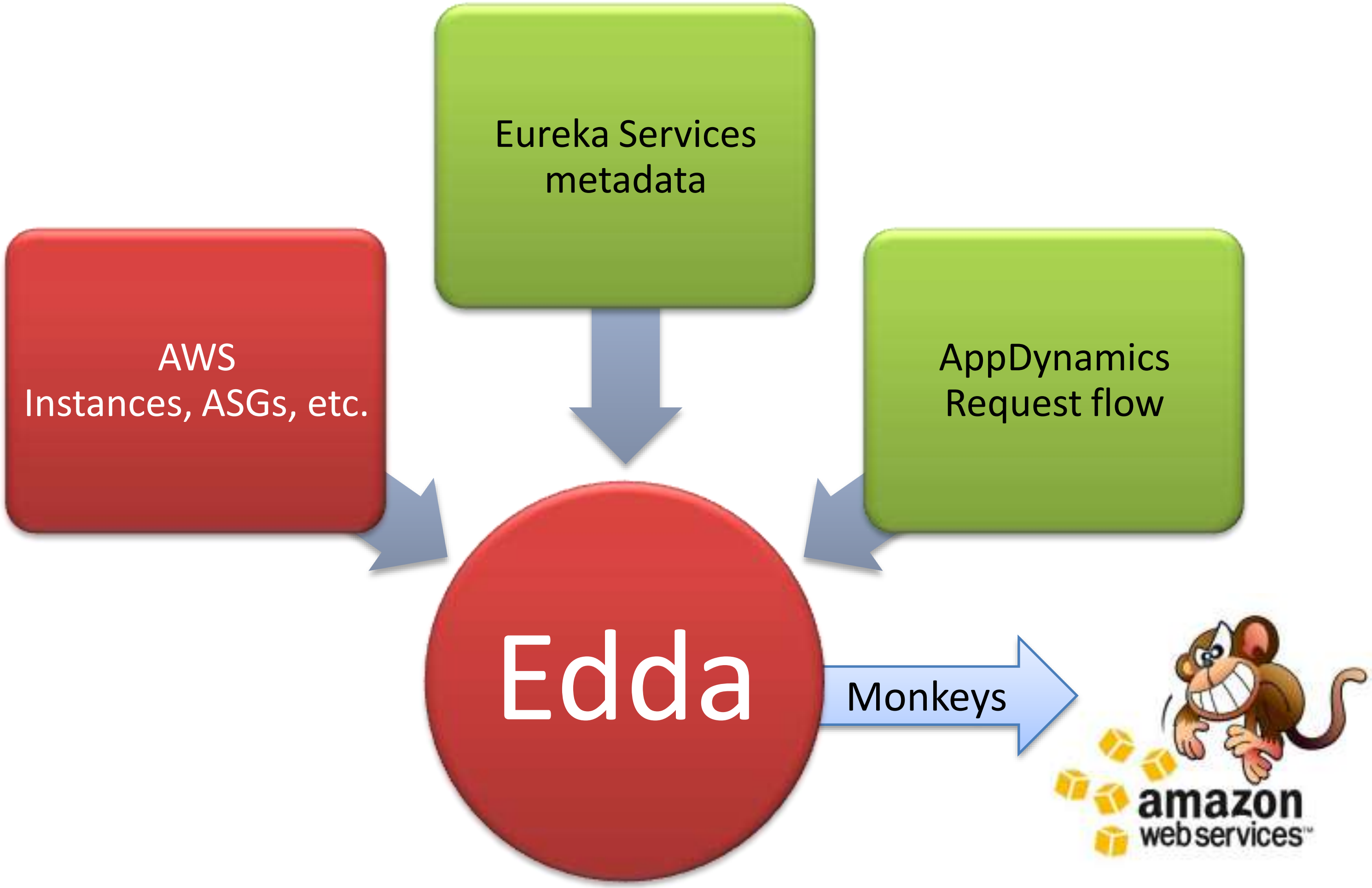
Chaos Monkey

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

- Computers (Datacenter or AWS) randomly die
  - Fact of life, but too infrequent to test resiliency
- Test to make sure systems are resilient
  - Kill individual instances without customer impact
- Latency Monkey (coming soon)
  - Inject 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"
…
 }
```

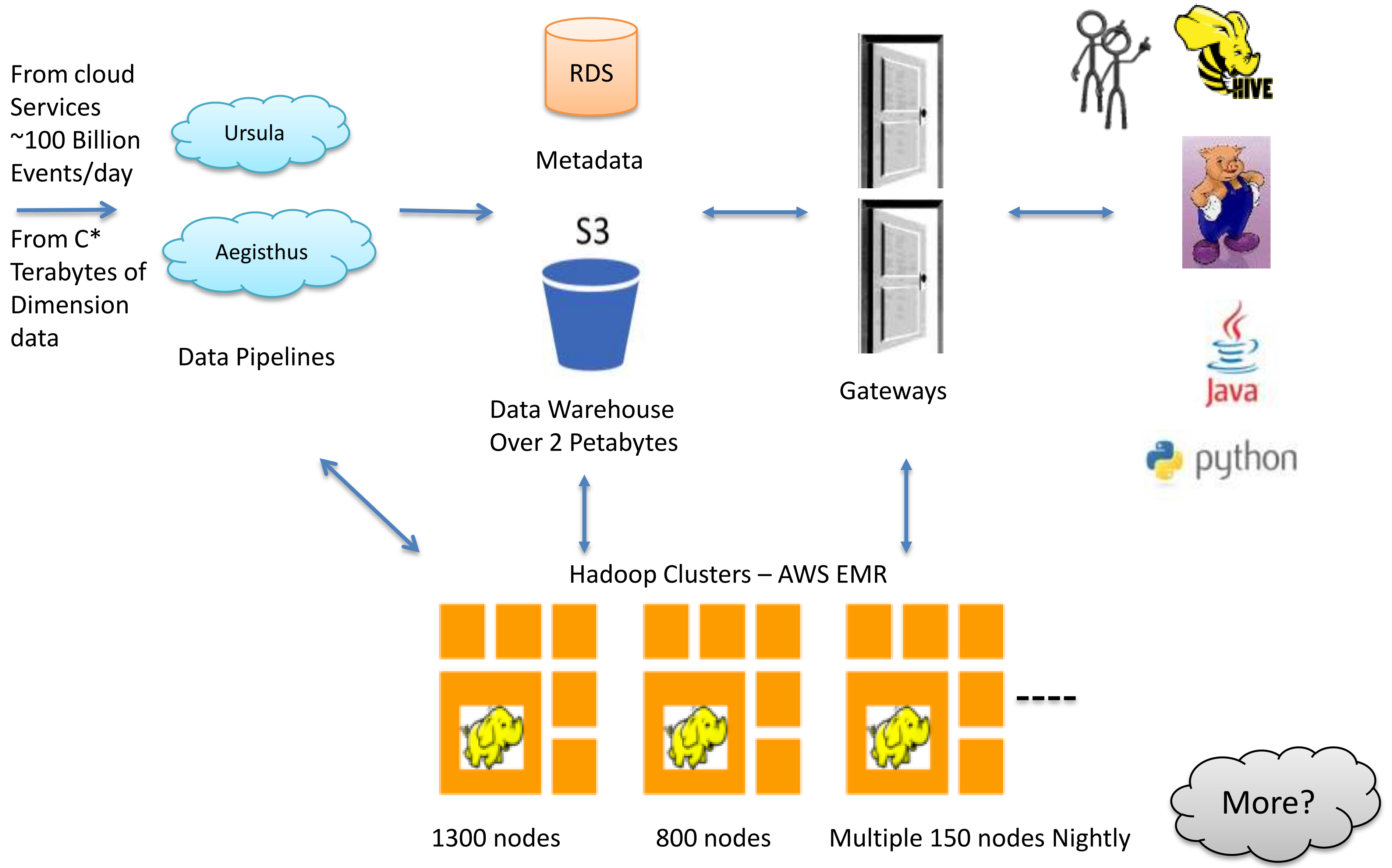
Cloud Native Big Data

Size the cluster to the data

Size the cluster to the questions

Never wait for space or answers

Netflix Dataoven


From cloud Services ~100 Billion Events/day
From C* Terabytes of Dimension data
Ursula
Aegisthus
Data Pipelines
RDS
Metadata
S3
Data Warehouse Over 2 Petabytes
Gateways
HIVE
Java
python
Hadoop Clusters – AWS EMR

1300 nodes
800 nodes
Multiple 150 nodes Nightly
More?

Cloud Native Development Patterns

Master copies of data are cloud resident

Dynamically provisioned micro-services

Services are distributed and ephemeral

Datacenter to Cloud Transition Goals

- Faster
  - **Lower latency** than the equivalent datacenter web pages and API calls
  - Measured as mean and 99th percentile
  - For both first hit (e.g. home page) and in-session hits for the same user
- Scalable
  - **Avoid needing any more datacenter capacity** as subscriber count increases
  - No central vertically scaled databases
  - Leverage AWS elastic capacity effectively
- Available
  - Substantially **higher robustness and availability** than datacenter services
  - Leverage multiple AWS availability zones
  - No scheduled down time, no central database schema to change
- Productive
  - Optimize **agility** of a large development team with automation and tools
  - Leave behind complex tangled datacenter code base (~8 year old architecture)
  - Enforce clean layered interfaces and re-usable components

Datacenter Anti-Patterns

What do we currently do in the datacenter that prevents us from meeting our goals?

Rewrite from Scratch

Not everything is cloud specific

Pay down technical debt

Robust patterns

Netflix Datacenter vs. Cloud Arch

Anti-Architecture

Central SQL Database	Distributed Key/Value NoSQL
Sticky In-Memory Session	Shared Memcached Session
Chatty Protocols	Latency Tolerant Protocols
Tangled Service Interfaces	Layered Service Interfaces
Instrumented Code	Instrumented Service Patterns
Fat Complex Objects	Lightweight Serializable Objects
Components as Jar Files	Components as Services

More?

Cloud Security

Fine grain security rather than perimeter

Leveraging AWS Scale to resist DDOS attacks

Automated attack surface monitoring and testing

http://www.slideshare.net/jason_chan/resilience-and-security-scale-lessons-learned

Security Architecture

- Instance Level Security baked into base AMI
  - Login: ssh only allowed via portal (not between instances)
  - Each app type runs as its own userid app{test|prod}
- AWS Security, Identity and Access Management
  - Each app has its own security group (firewall ports)
  - Fine grain user roles and resource ACLs
- Key Management
  - AWS Keys dynamically provisioned, easy updates
  - High grade app specific key management using HSM

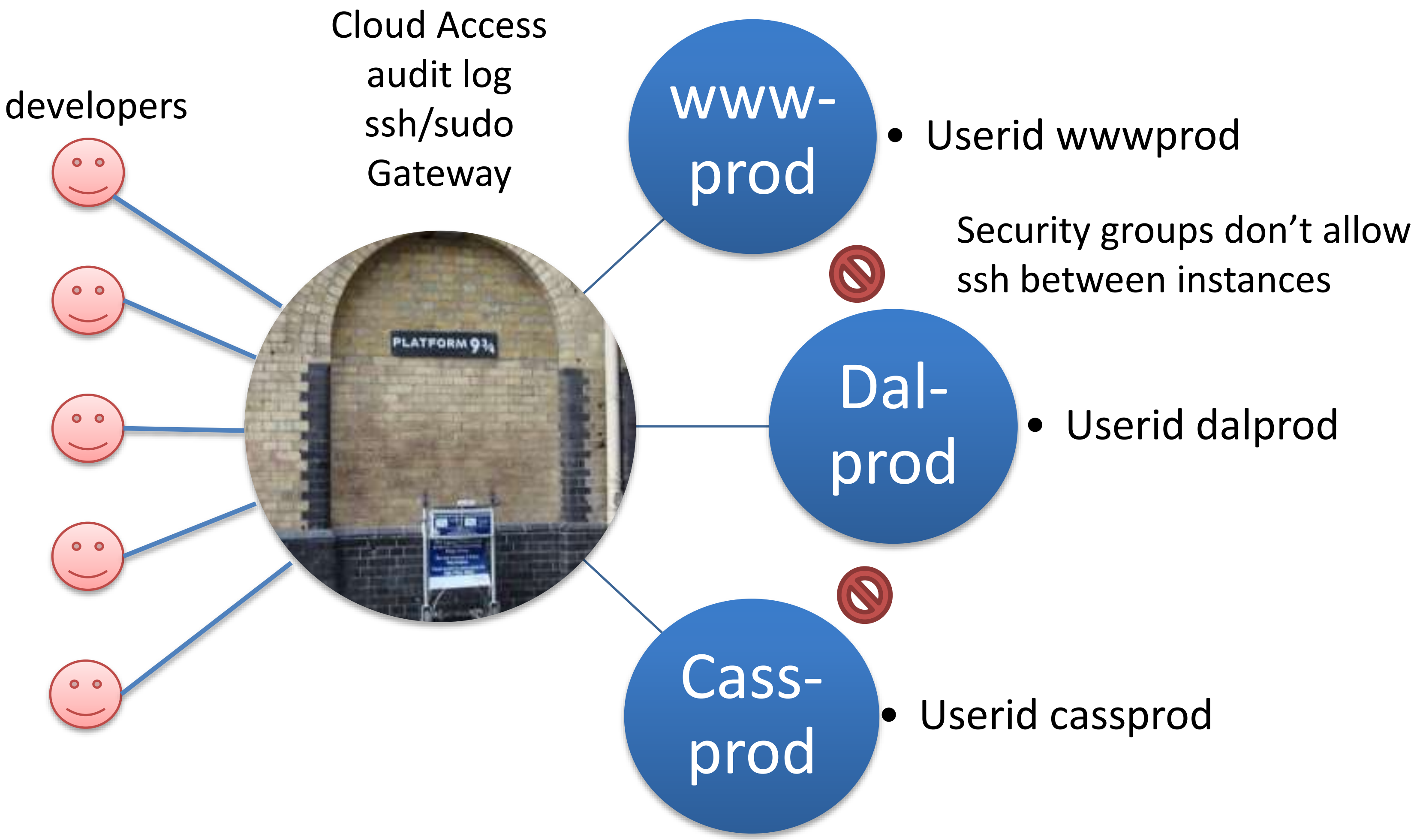
More?

AWS Accounts

Accounts Isolate Concerns

- paastest – for development and testing
  - Fully functional deployment of all services
  - Developer tagged “stacks” for separation
- paasprod – for production
  - Autoscale groups only, isolated instances are terminated
  - Alert routing, backups enabled by default
- paasaudit – for sensitive services
  - To support SOX, PCI, etc.
  - Extra access controls, auditing
- paasarchive – for disaster recovery
  - Long term archive of backups
  - Different region, perhaps different vendor

Cloud Access Control



- Userid wwwprod

Security groups don't allow ssh between instances

- Userid dalprod

- Userid cassprod

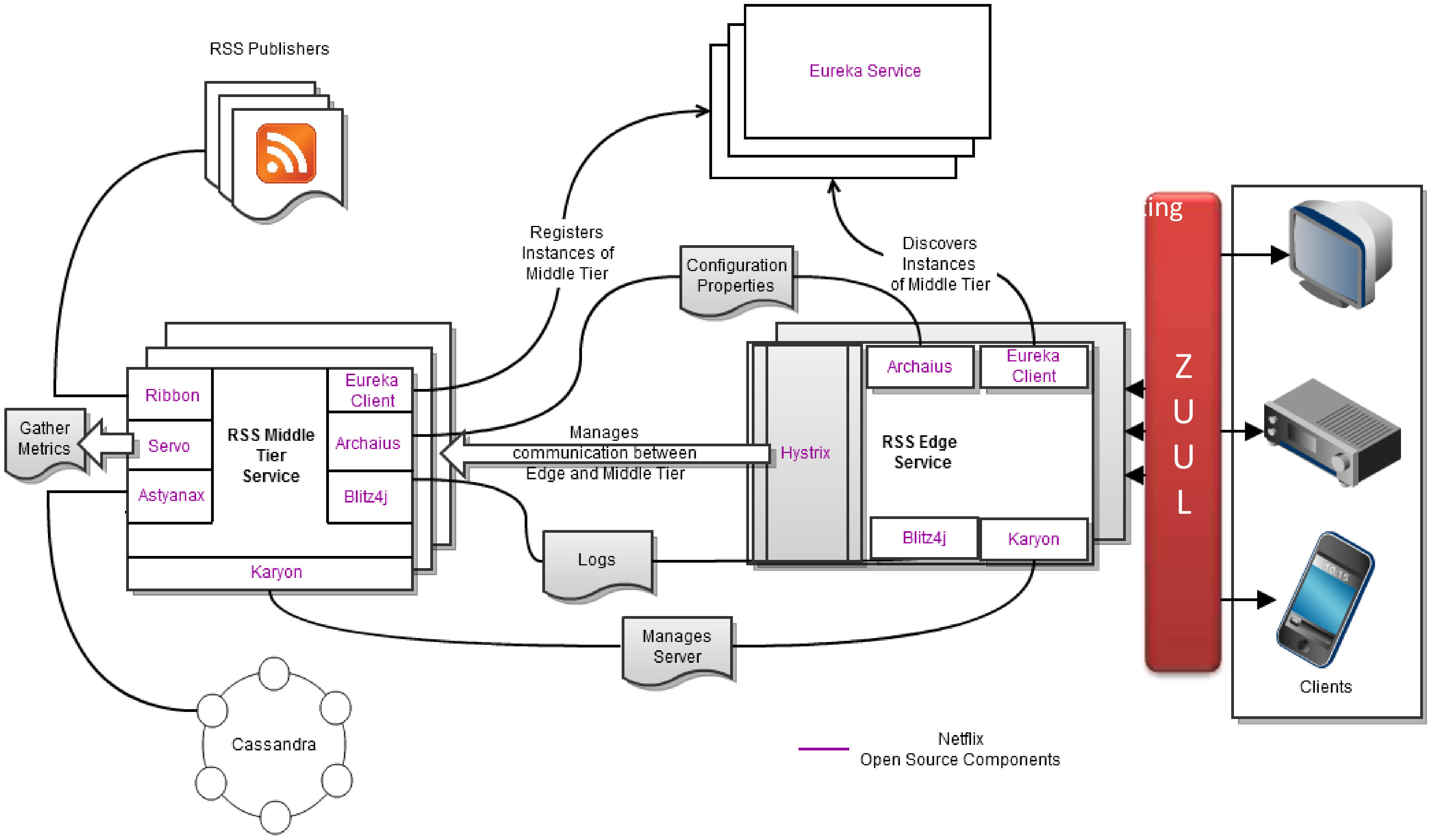
NETFLIX OSS

Our perspiration…

A Cloud Native Open Source Platform

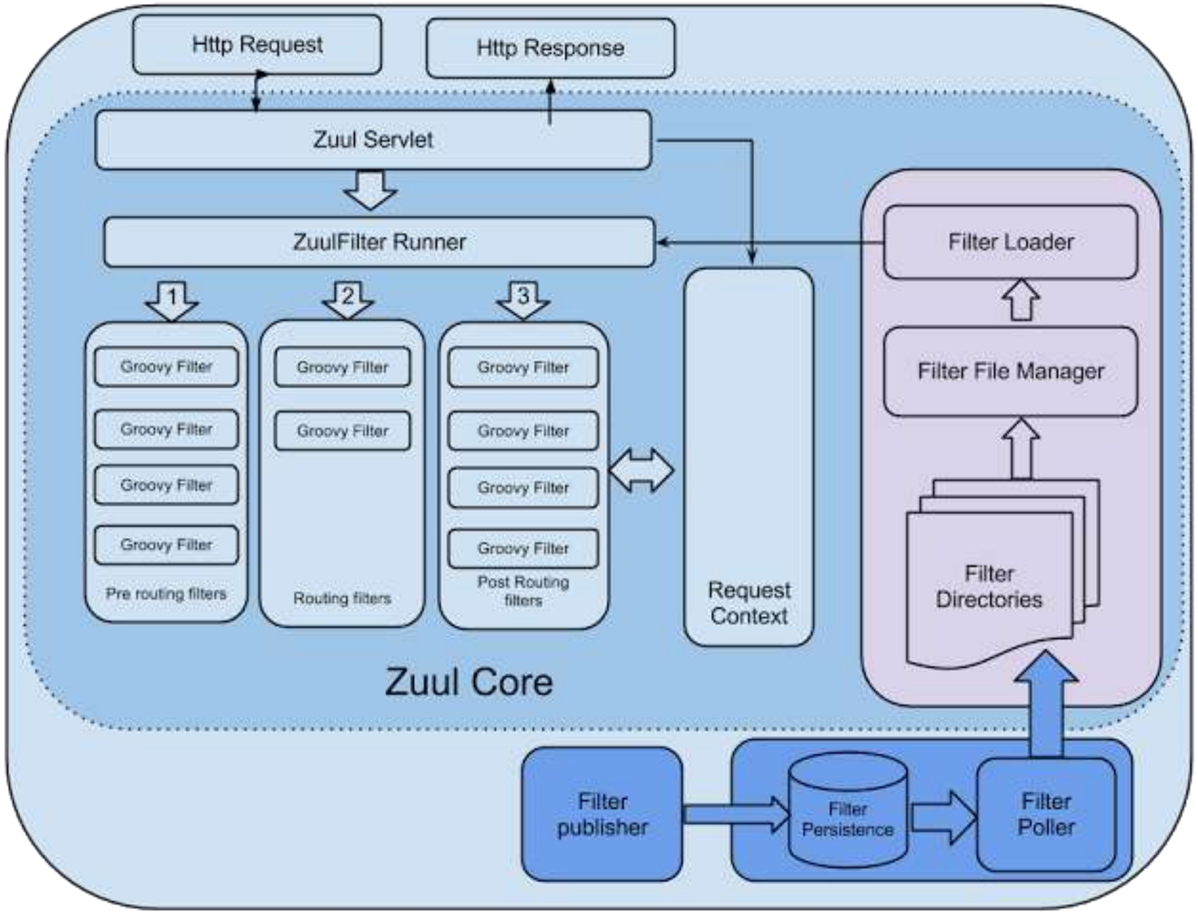
See netflix.github.com

Example Application – RSS Reader



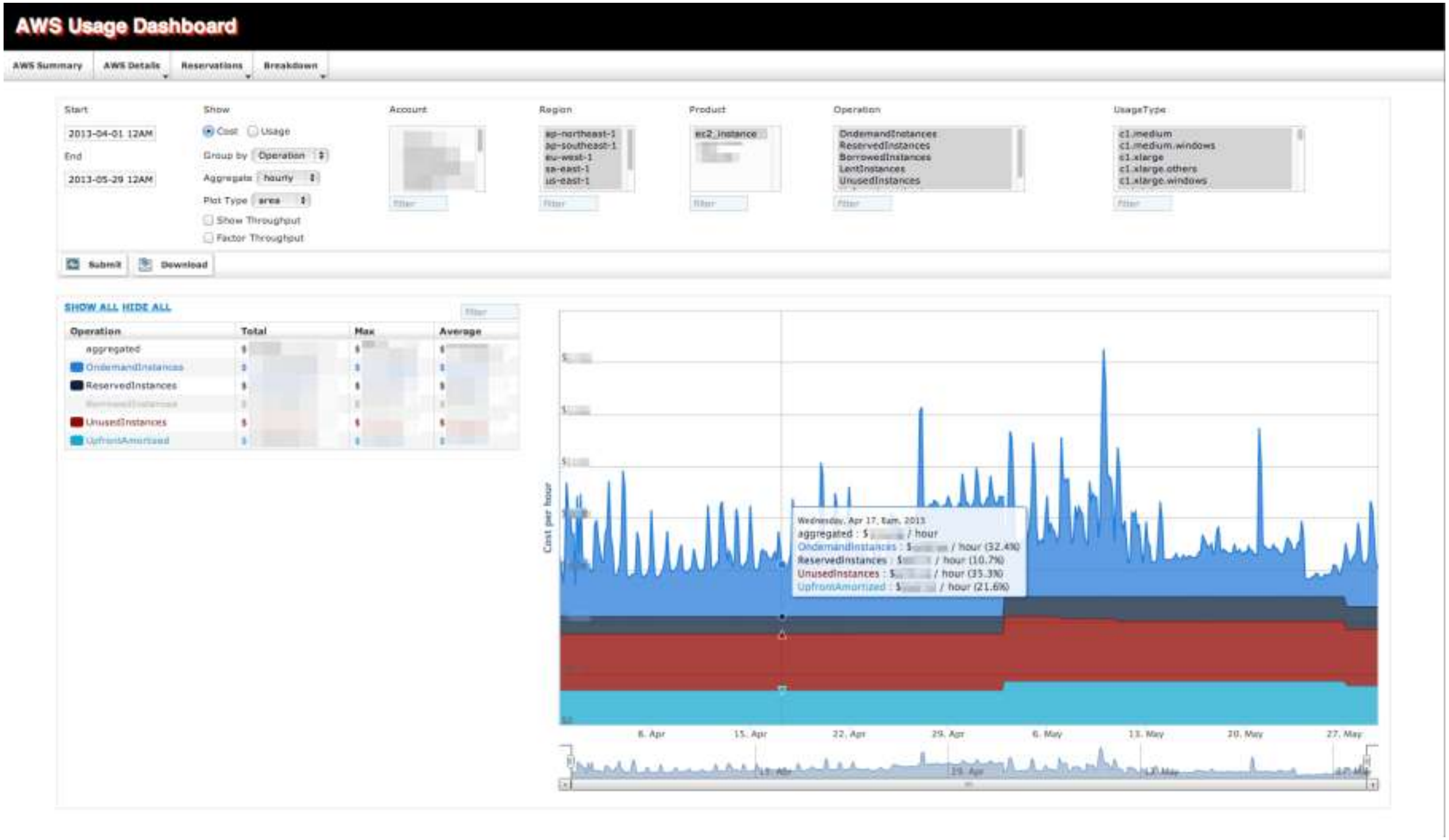
Zuul Architecture

http://techblog.netflix.com/2013/06/announcing-zuul-edge-service-in-cloud.html



Ice – AWS Usage Tracking

http://techblog.netflix.com/2013/06/announcing-ice-cloud-spend-and-usage.html



Start: 2013-04-01 12AM
End: 2013-05-29 12AM

Show: [✓ Cost] Usage
Group by: Operation
Aggregate: hourly
Plot Type: area
☐ Show Throughput
☐ Factor Throughput

Account: [redacted]

Region: ap-northeast-1, ap-southeast-1, eu-west-1, sa-east-1, us-east-1

Product: ec2_instance

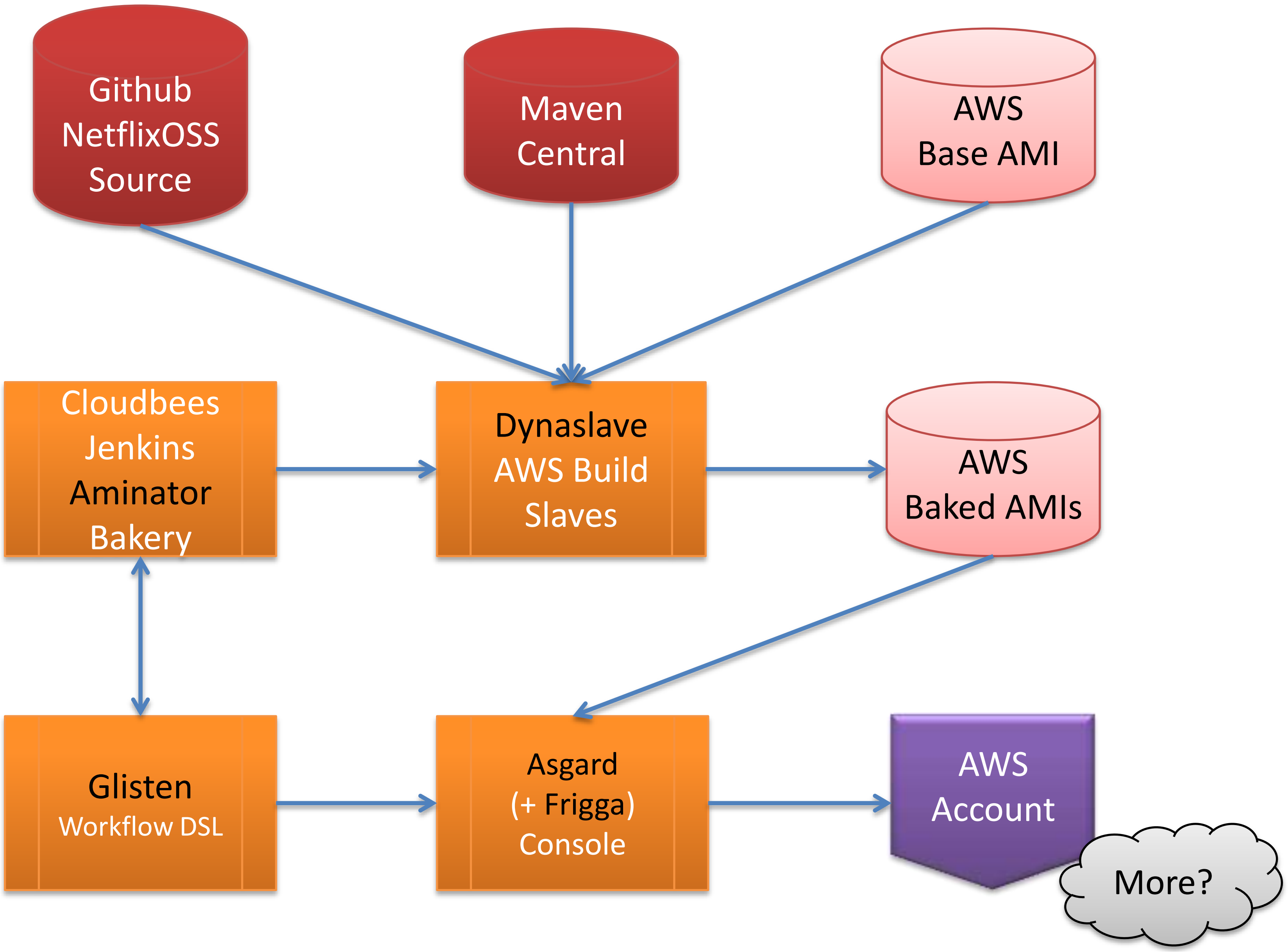
Operation: OndemandInstances, ReservedInstances, BorrowedInstances, LentInstances, UnusedInstances

UsageType: c1.medium, c1.medium.windows, c1.xlarge, c1.xlarge.others, c1.xlarge.windows

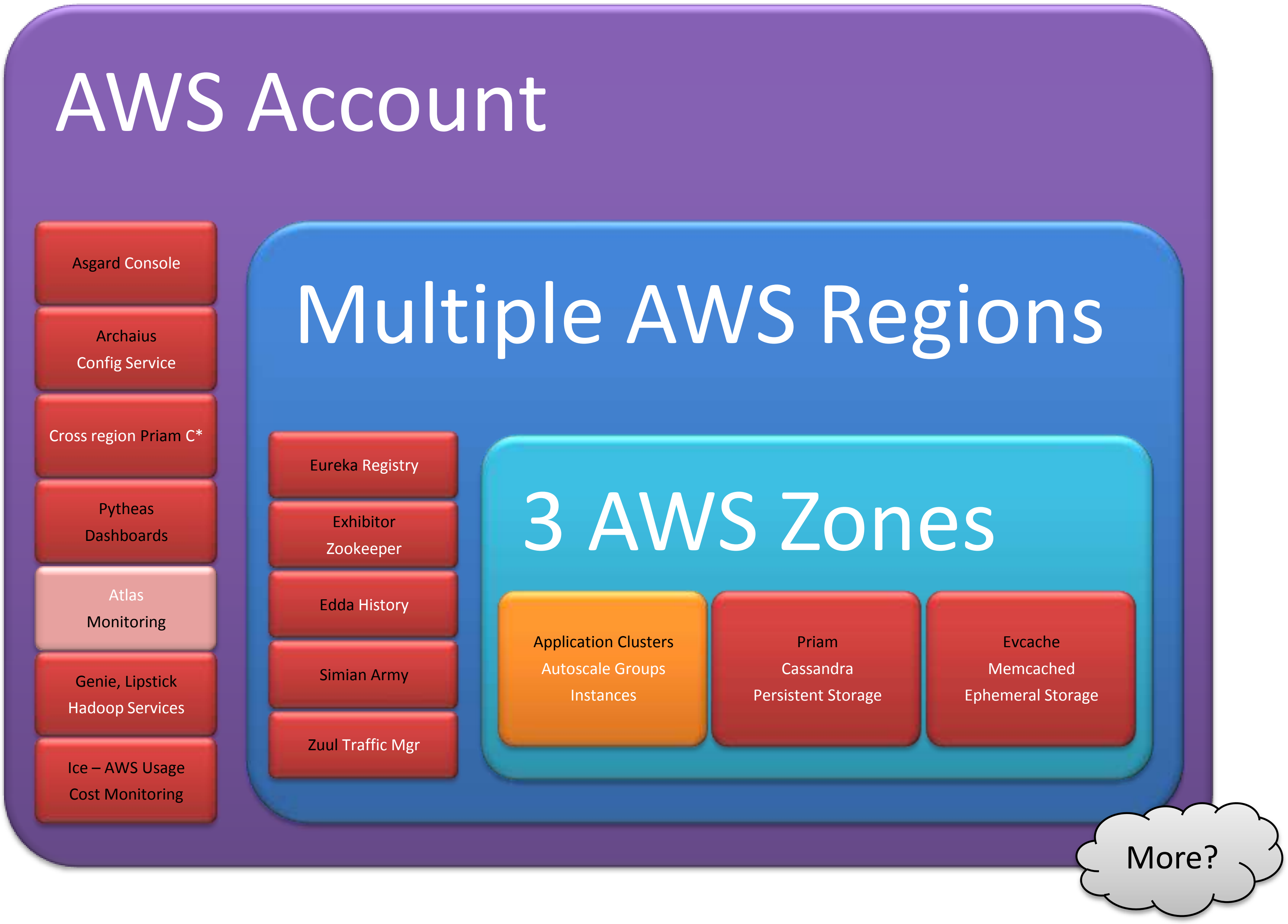
Operation	Total	Max	Average
aggregated	$[redacted]	$[redacted]	$[redacted]
🟦 OndemandInstances	$[redacted]	$[redacted]	$[redacted]
⬛ ReservedInstances	$[redacted]	$[redacted]	$[redacted]
BorrowedInstances	$[redacted]	$[redacted]	$[redacted]
🟥 UnusedInstances	$[redacted]	$[redacted]	$[redacted]
🟦 UpfrontAmortized	$[redacted]	$[redacted]	$[redacted]

Wednesday, Apr 17, 8am, 2013
aggregated : $[redacted] / hour
OndemandInstances : $[redacted] / hour (32.4%)
ReservedInstances : $[redacted] / hour (10.7%)
UnusedInstances : $[redacted] / hour (35.3%)
UpfrontAmortized : $[redacted] / hour (21.6%)

NetflixOSS Continuous Build and Deployment



NetflixOSS Services Scope



NetflixOSS Instance Libraries

Initialization

- Baked AMI – Tomcat, Apache, your code
- Governator – 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 and Feign - REST Clients 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

More?

NetflixOSS Testing and Automation

Test Tools

- CassJmeter – Load testing for Cassandra
- Circus Monkey – Test account reservation rebalancing

Maintenance

- Janitor Monkey – Cleans up unused resources
- Efficiency Monkey
- Doctor Monkey
- Howler Monkey – Complains about AWS limits

Availability

- Chaos Monkey – Kills Instances
- Chaos Gorilla – Kills Availability Zones
- Chaos Kong – Kills Regions
- Latency Monkey – Latency and error injection

Security

- Conformity Monkey – architectural pattern warnings
- Security Monkey – security group and S3 bucket permissions

More?

Vendor Driven Portability

Interest in using NetflixOSS for Enterprise Private Clouds

"It's done when it runs Asgard"
Functionally complete
Demonstrated March
Released June in V3.3

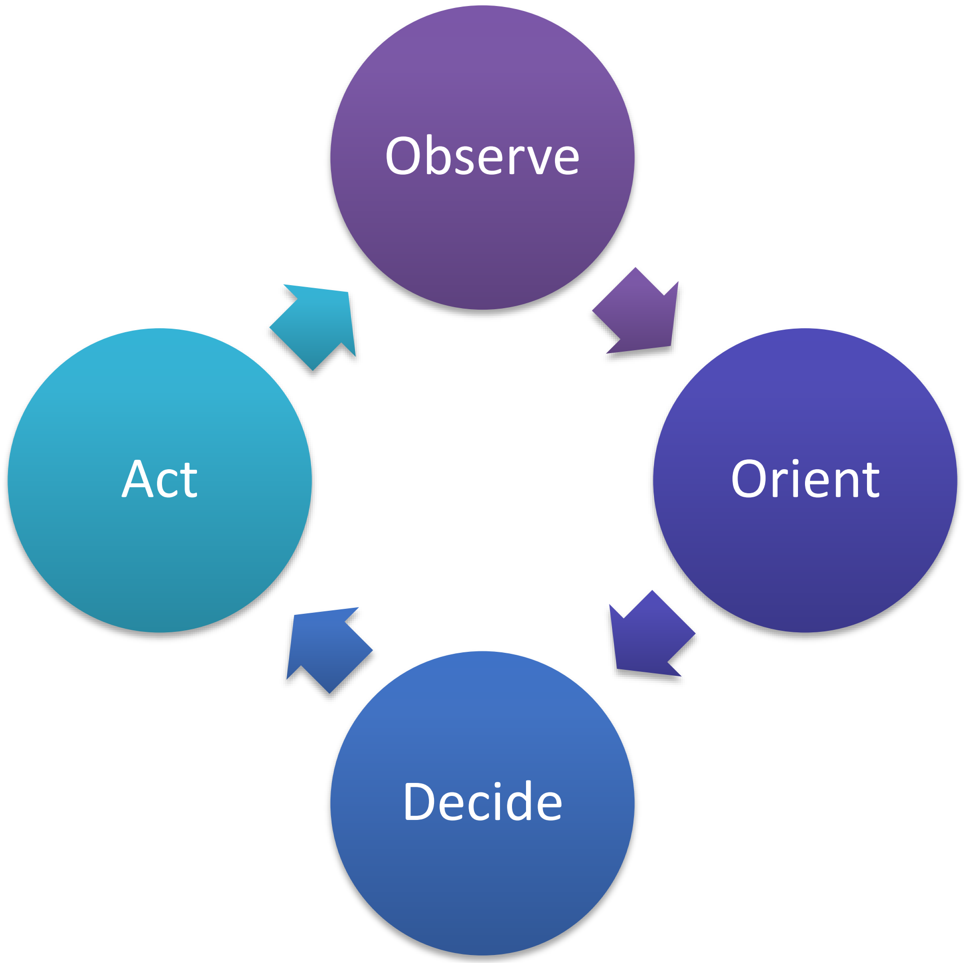
IBM Example application "Acme Air"
Based on NetflixOSS running on AWS
Ported to IBM Softlayer with Rightscale

Vendor and end user interest
Openstack "Heat" getting there
Paypal C3 Console based on Asgard

Cost-Aware Cloud Architectures

Jinesh Varia
@jinman
Technology Evangelist

Adrian Cockcroft
@adrianco
Director, Architecture


Observe
Orient
Decide
Act

Experiment Often & Adapt Quickly

- Cost of failure falls dramatically
- Return on (small incremental) Investments is high
- More risk taking, more innovation
- More iteration, faster innovation

« Want to increase innovation? **Lower the cost of failure** »

Joi Ito

Accelerate building a new line of business

NASDAQ

Market Replay

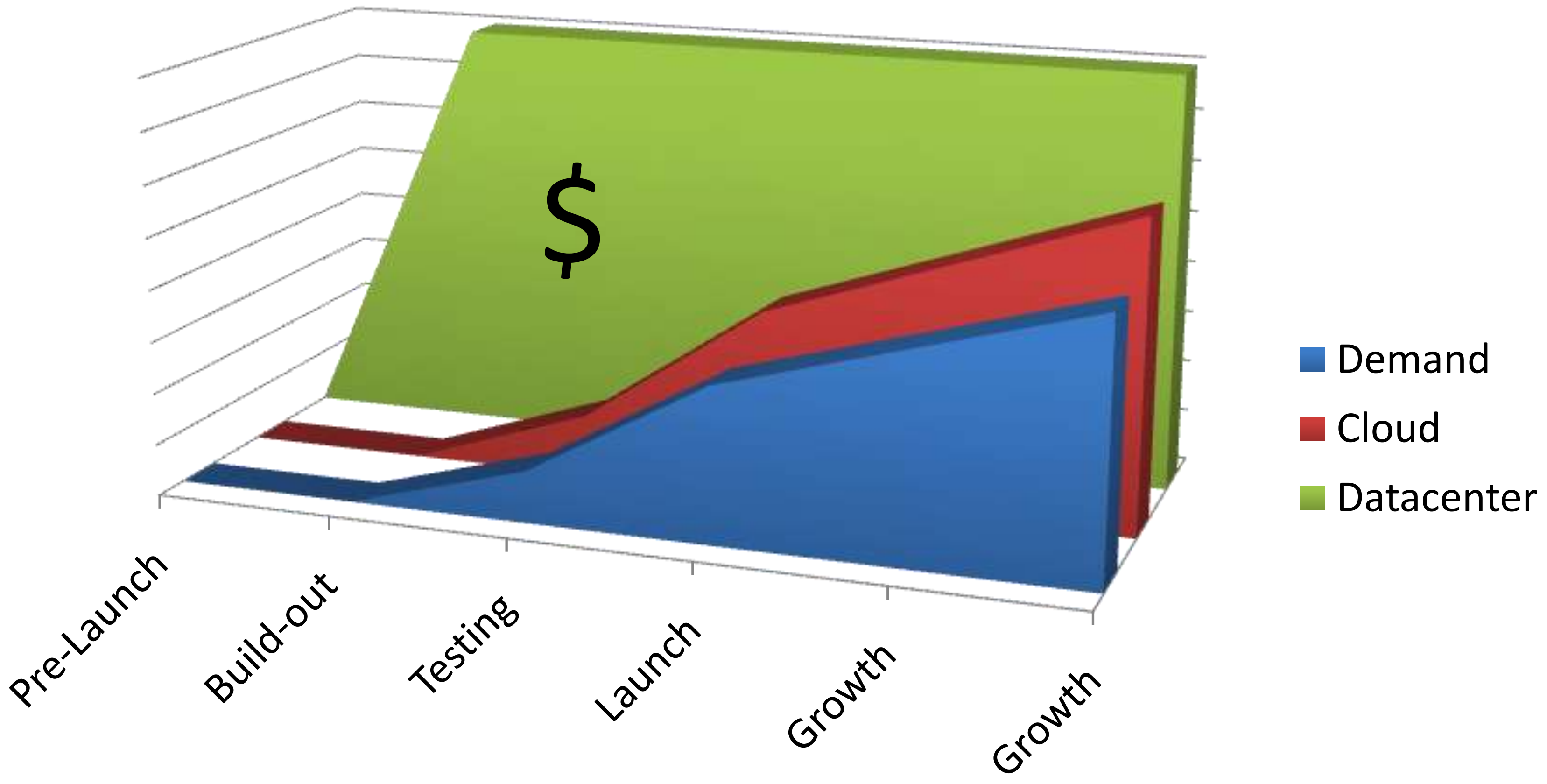
NASDAQ OMX | FinQloud
POWERED BY AWS

Go Global in Minutes

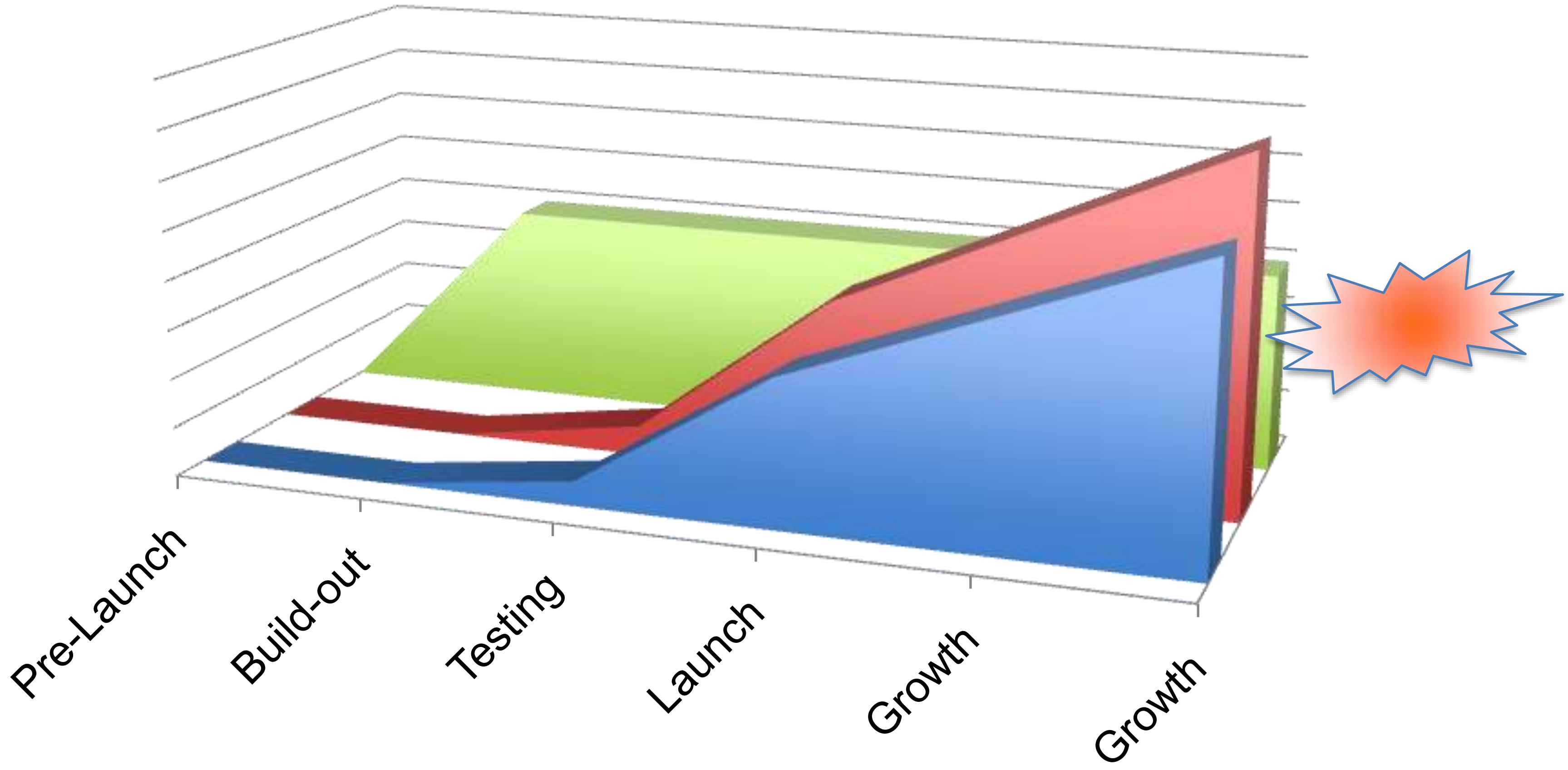
Netflix Examples

- European Launch using AWS Ireland
  - No employees in Ireland, no provisioning delay, everything worked
  - No need to do detailed capacity planning
  - Over-provisioned on day 1, shrunk to fit after a few days
  - Capacity grows as needed for additional country launches
- Brazilian Proxy Experiment
  - No employees in Brazil, no “meetings with IT”
  - Deployed instances into two zones in AWS Brazil
  - Experimented with network proxy optimization
  - Decided that gain wasn’t enough, shut everything down

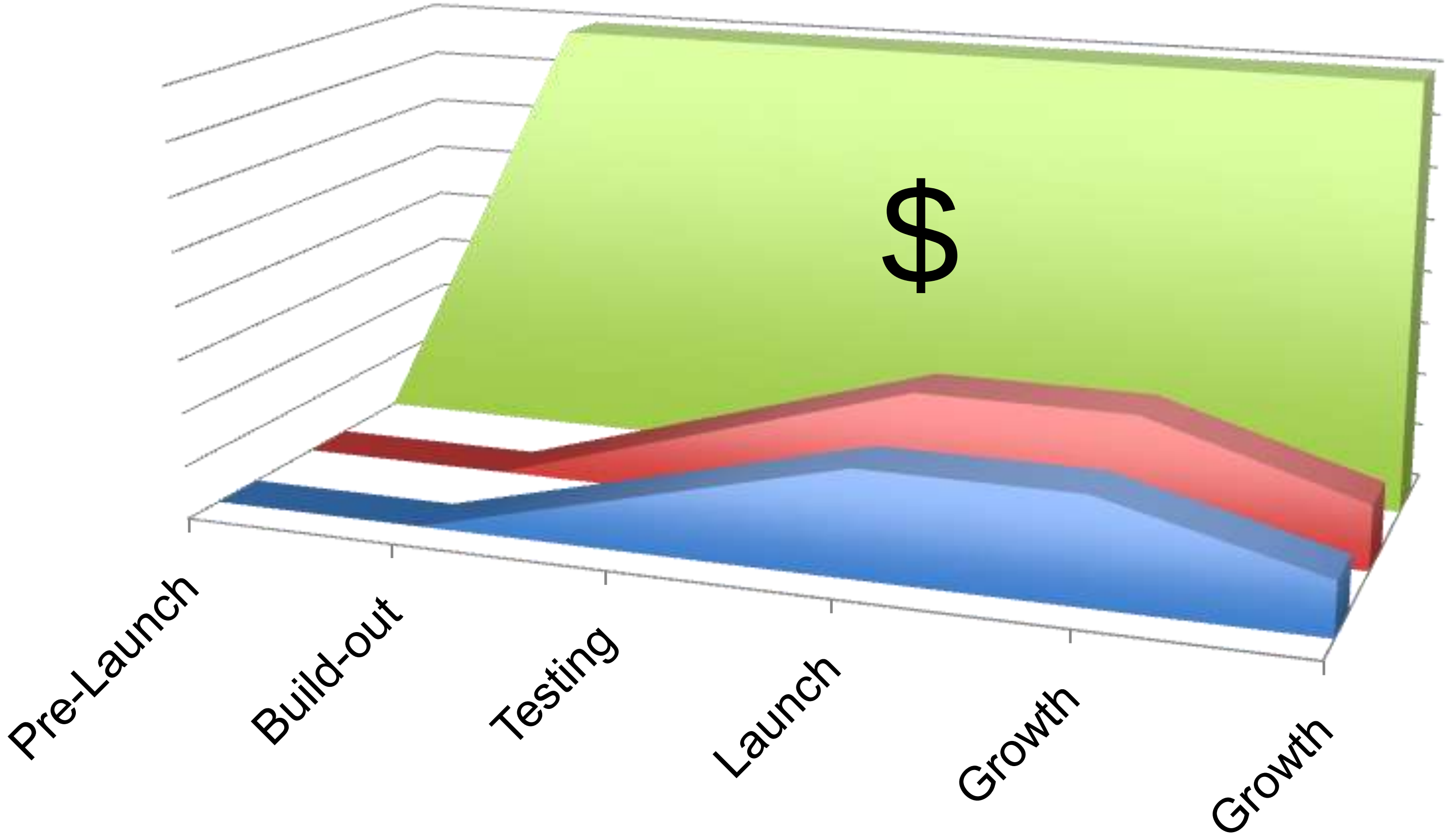
Product Launch Agility - Rightsized



Product Launch - Under-estimated



Product Launch Agility – Over-estimated

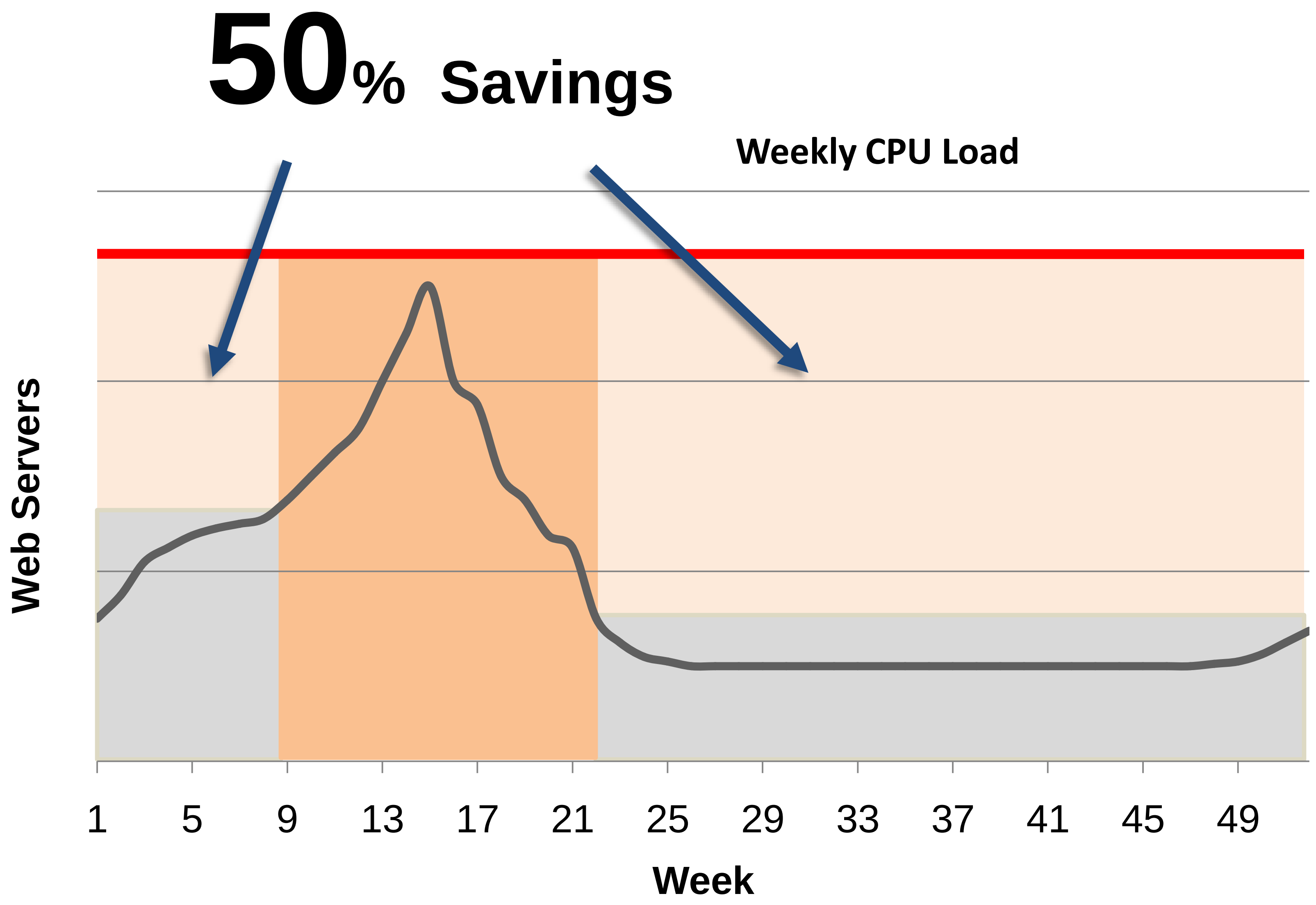


Return on Agility (Agile ROI) = More Revenue

Key Takeaways on Cost-Aware Architectures….

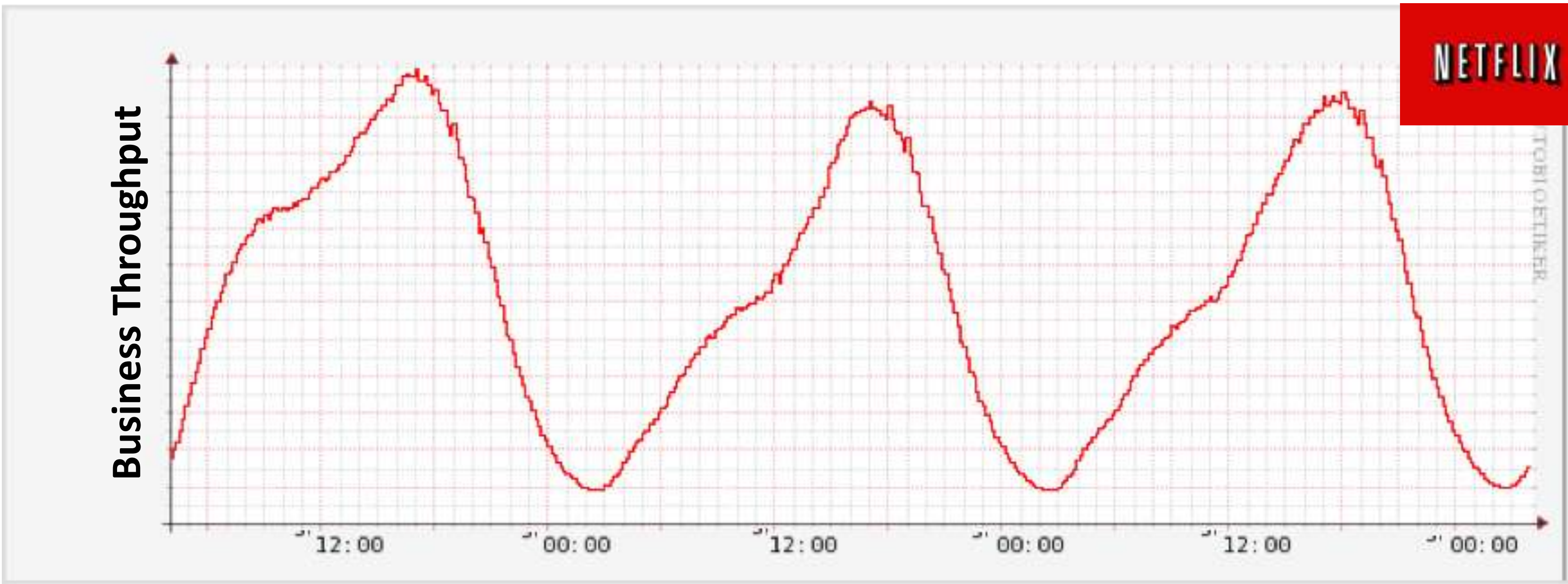
#1 Business Agility by Rapid Experimentation = Increased Revenue

When you turn off your cloud resources, you actually **stop paying for them**



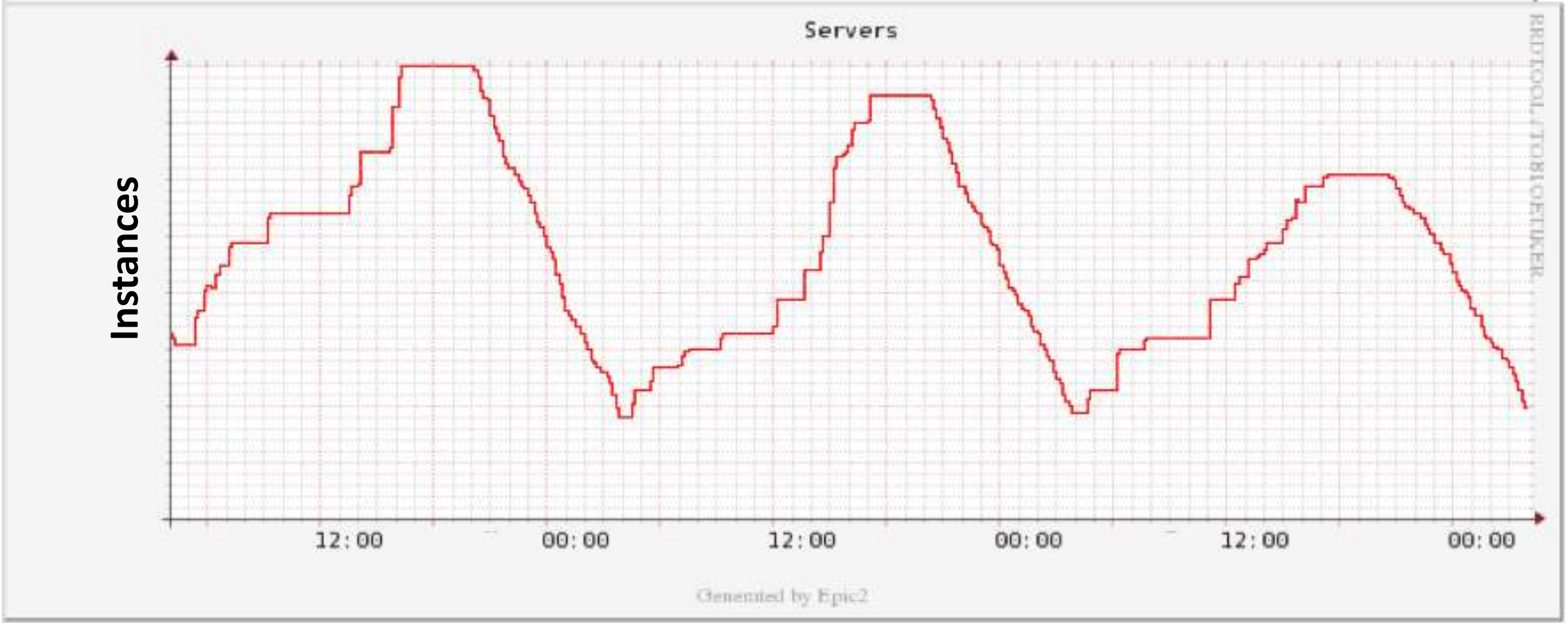
Optimize during a year

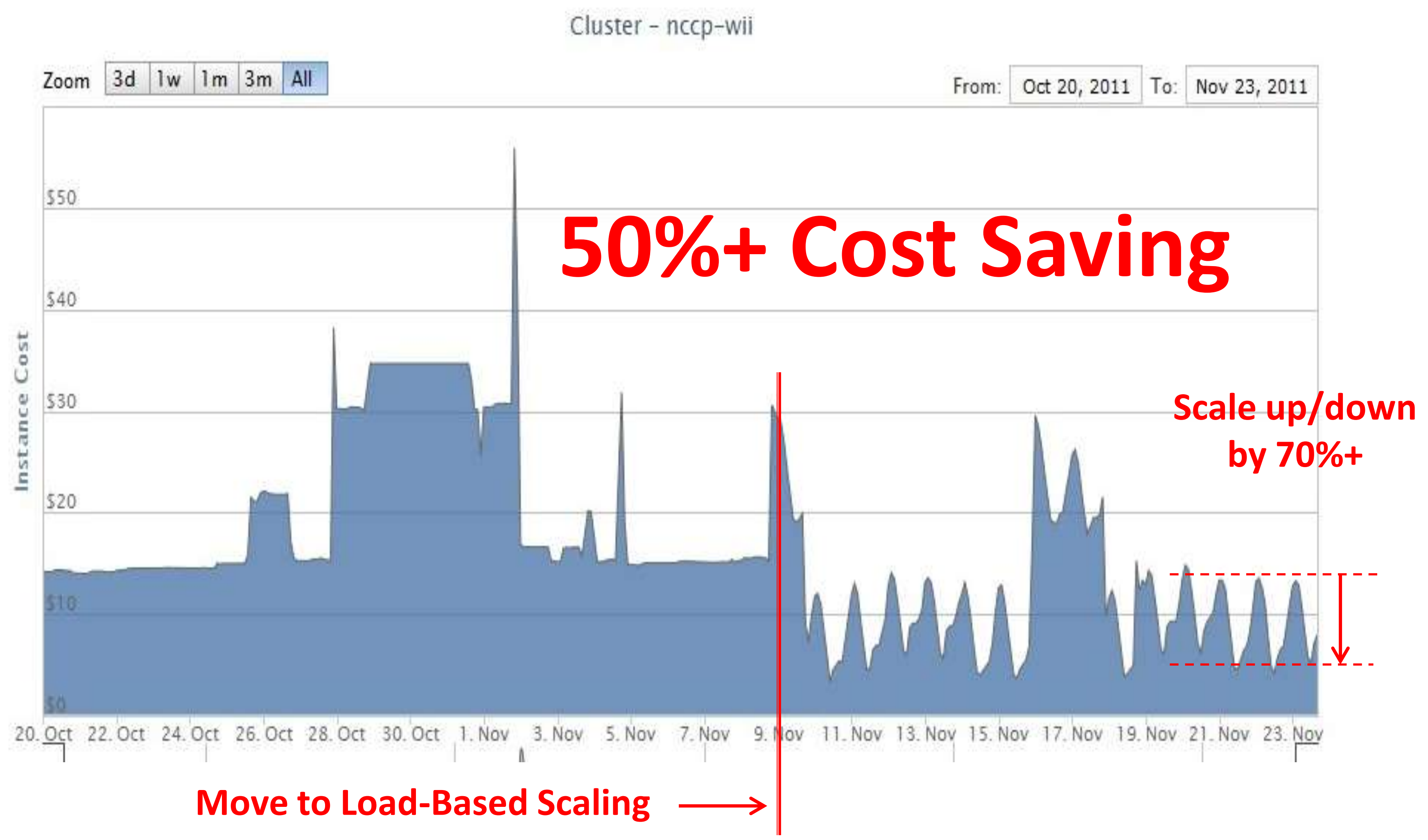
Business Throughput



NETFLIX

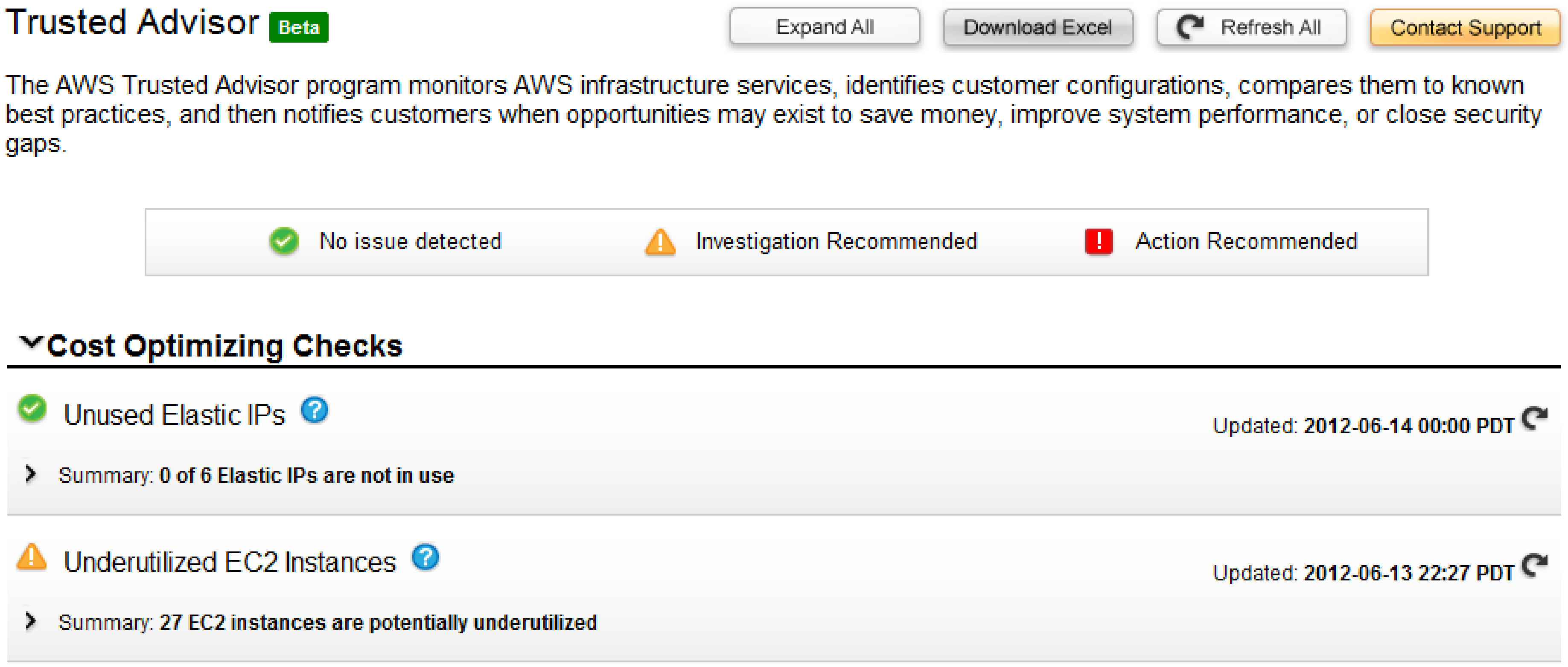
Instances




Cluster - nccp-wii
Zoom
3d
1w
1m
3m
All
From: Oct 20, 2011
To: Nov 23, 2011
Instance Cost
$50
$40
$30
$20
$10
$0
20. Oct
22. Oct
24. Oct
26. Oct
28. Oct
30. Oct
1. Nov
3. Nov
5. Nov
7. Nov
9. Nov
11. Nov
13. Nov
15. Nov
17. Nov
19. Nov
21. Nov
23. Nov
50%+ Cost Saving
Scale up/down by 70%+
Move to Load-Based Scaling

Pay as you go

AWS Support – Trusted Advisor – Your personal cloud assistant



The AWS Trusted Advisor program monitors AWS infrastructure services, identifies customer configurations, compares them to known best practices, and then notifies customers when opportunities may exist to save money, improve system performance, or close security gaps.

✓ No issue detected ⚠ Investigation Recommended ❗ Action Recommended

Cost Optimizing Checks

✓ Unused Elastic IPs — Updated: **2012-06-14 00:00 PDT**

Summary: **0 of 6 Elastic IPs are not in use**

⚠ Underutilized EC2 Instances — Updated: **2012-06-13 22:27 PDT**

Summary: **27 EC2 instances are potentially underutilized**

Other simple optimization tips

- **Don’t forget to…**
  - Disassociate unused EIPs
  - Delete unassociated Amazon EBS volumes
  - Delete older Amazon EBS snapshots
  - Leverage Amazon S3 Object Expiration

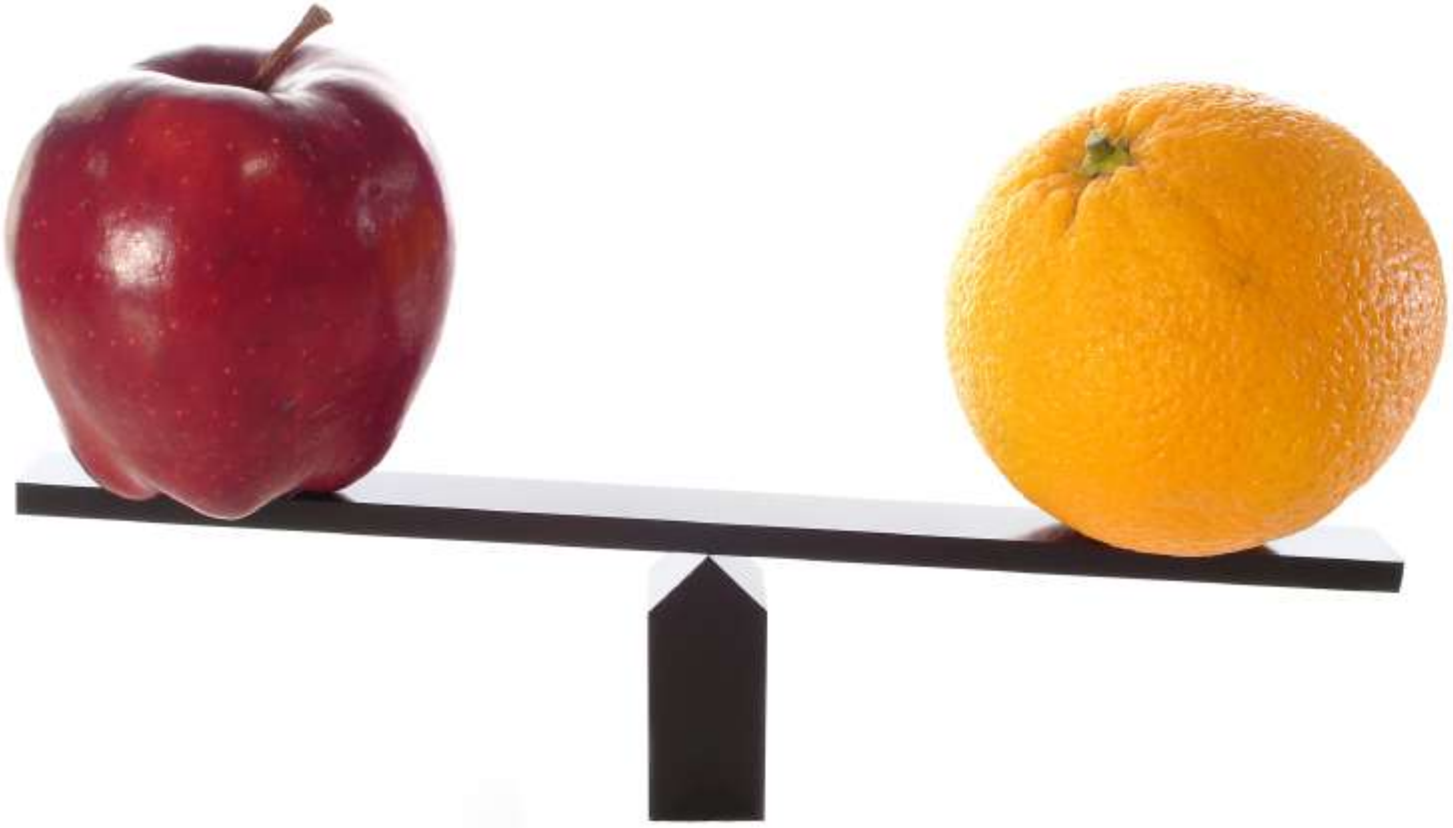
NETFLIX OSS Janitor Monkey cleans up unused resources

Building Cost-Aware Cloud Architectures

#1 Business Agility by Rapid Experimentation = Increased Revenue

#2 Business-driven Auto Scaling Architectures = Savings

When Comparing TCO…



When Comparing TCO…

Make sure that you are including all the cost factors into consideration

Place
Power
Pipes
People
Patterns

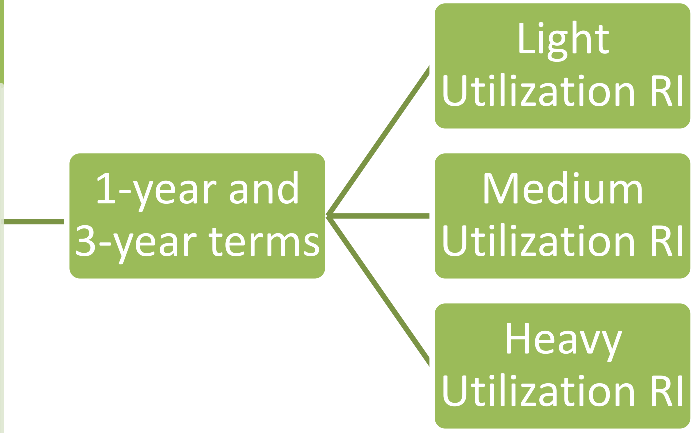
Save more when you reserve

On-demand Instances

- Pay as you go
- Starts from $0.02/Hour

Reserved Instances

- One time low upfront fee + Pay as you go
- $23 for 1 year term and $0.01/Hour



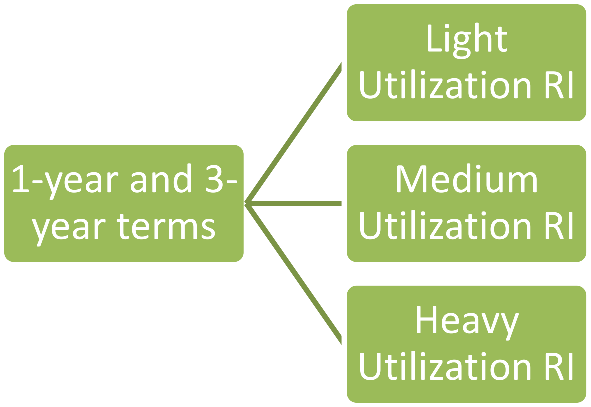
Break-even point

ed
es

ow
e + Pay

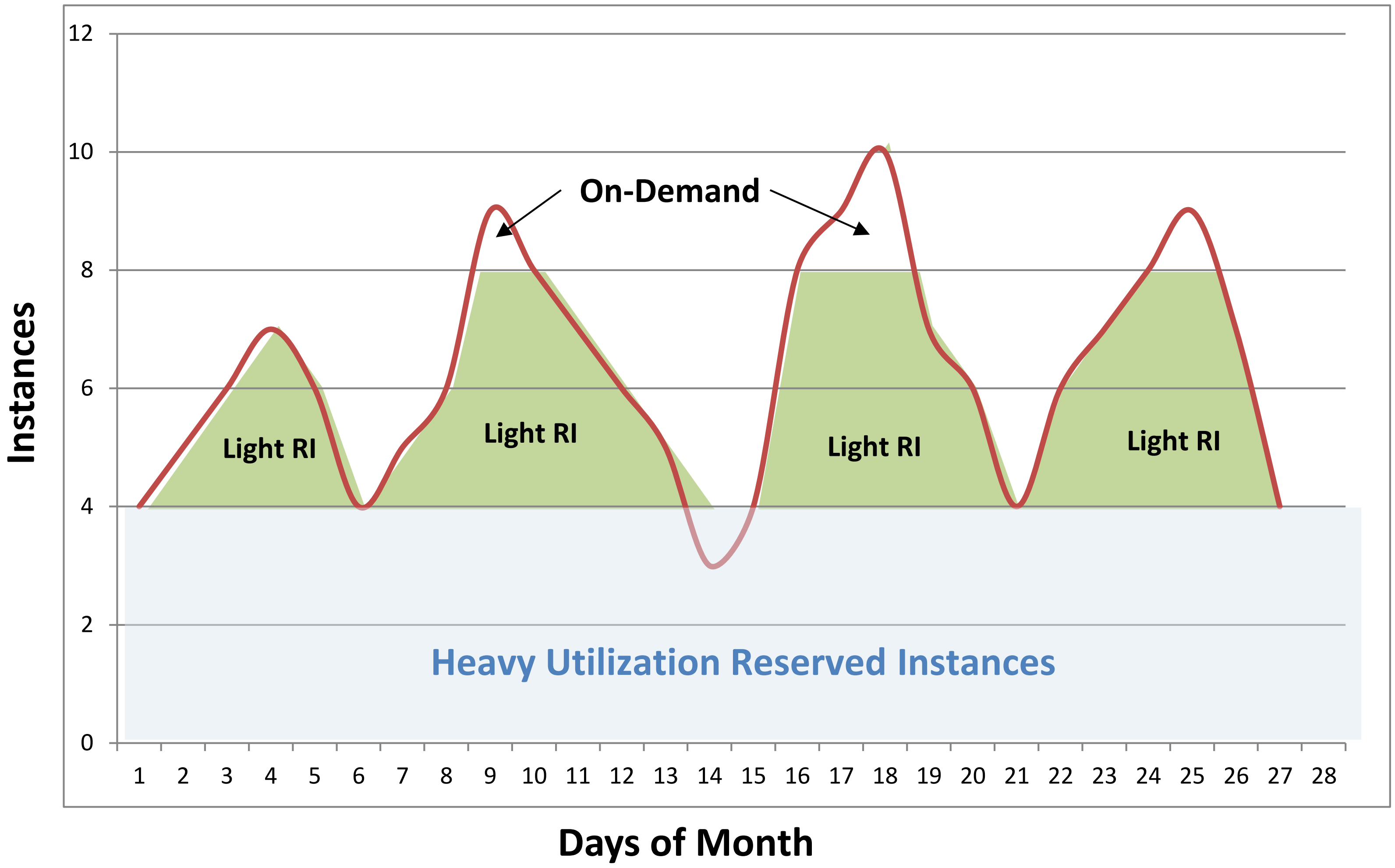
ear

ur

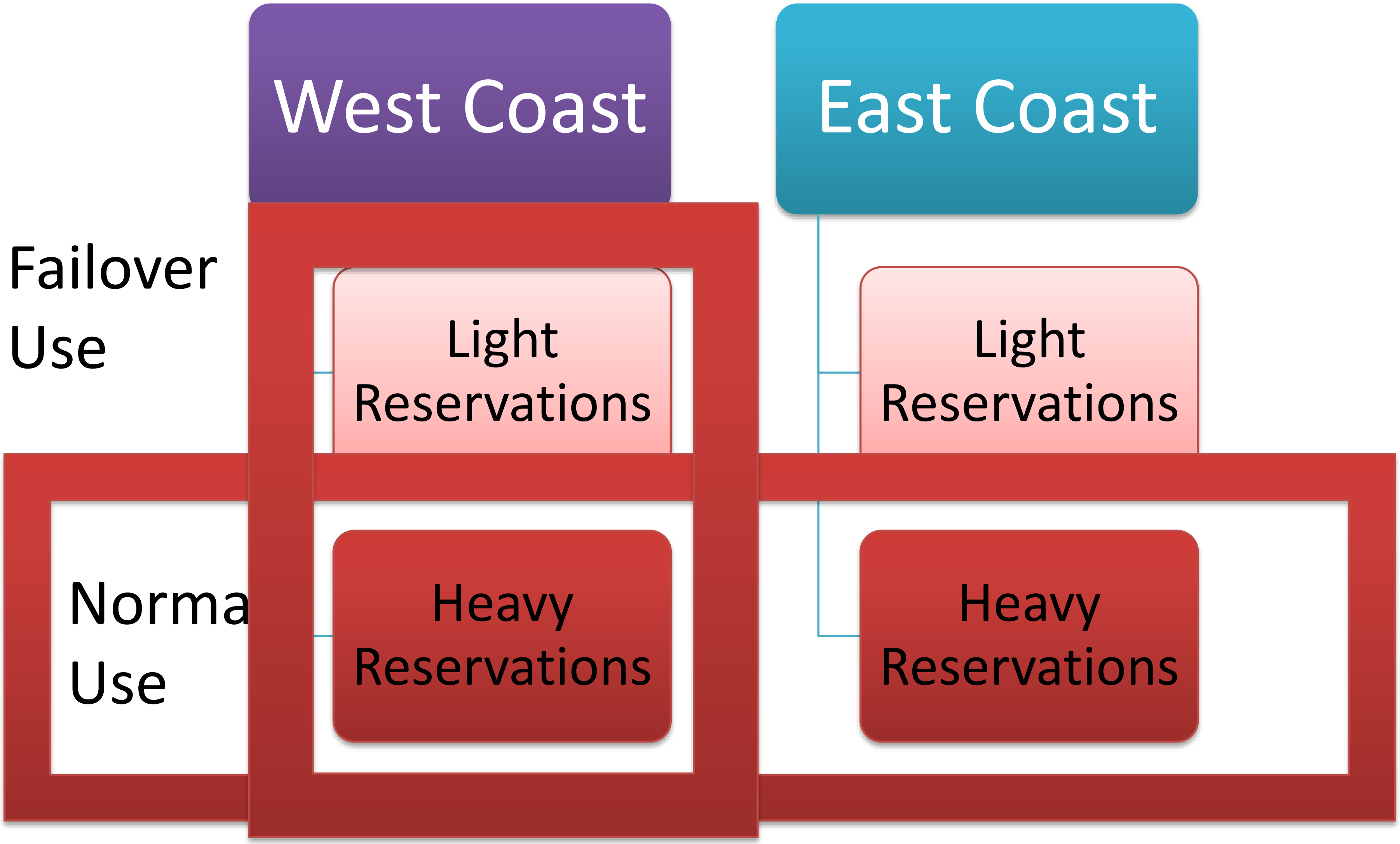


Utilization (Uptime)	Ideal For	Savings over On-Demand
10% - 40% **(>3.5 < 5.5 months/year)**	Disaster Recovery (Lowest Upfront)	**56%**
40% - 75% **(>5.5 < 7 months/year)**	Standard Reserved Capacity	**66%**
>75% **(>7 months/year)**	Baseline Servers (Lowest Total Cost)	**71%**

Mix and Match Reserved Types and On-Demand



Netflix Concept for Regional Failover Capacity



Building Cost-Aware Cloud Architectures

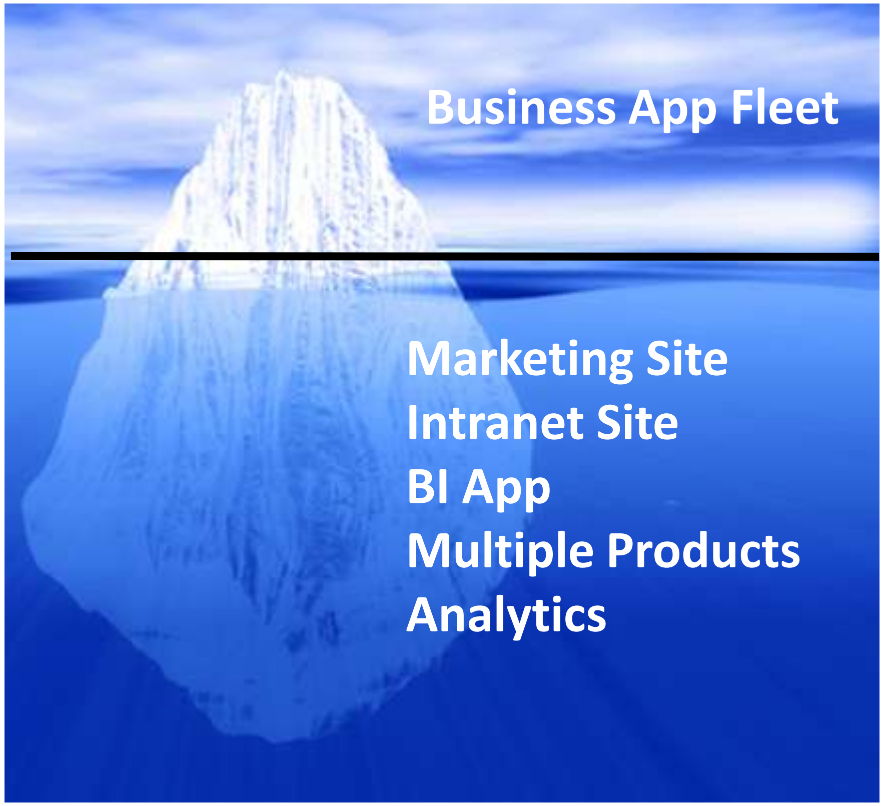
#1 Business Agility by Rapid Experimentation = Increased Revenue

#2 Business-driven Auto Scaling Architectures = Savings

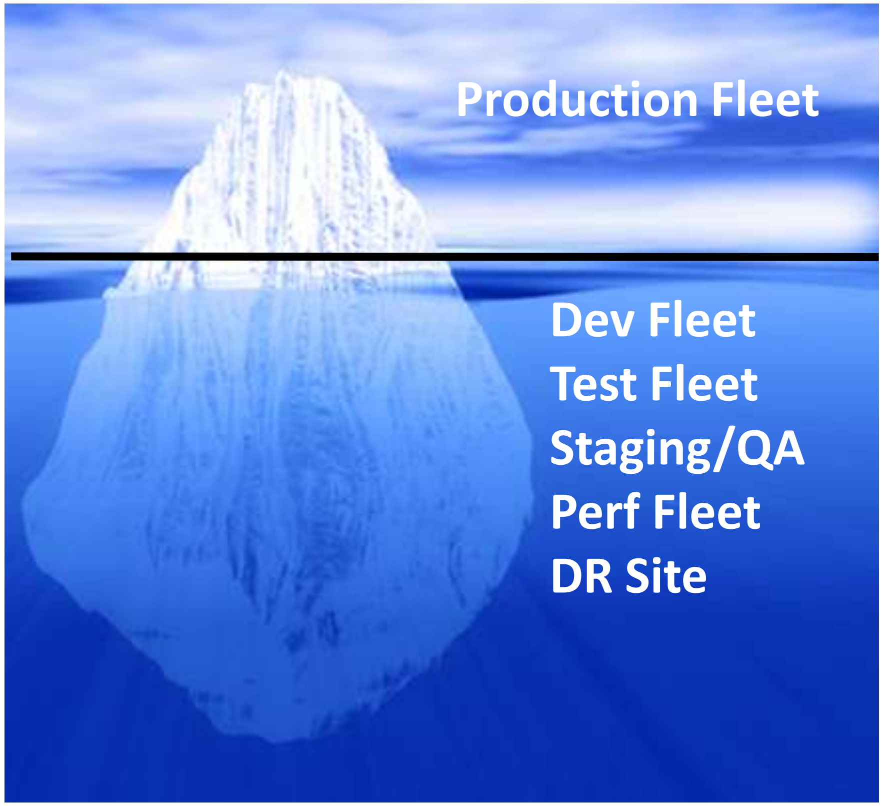
#3 Mix and Match Reserved Instances with On-Demand = Savings

Variety of Applications and Environments

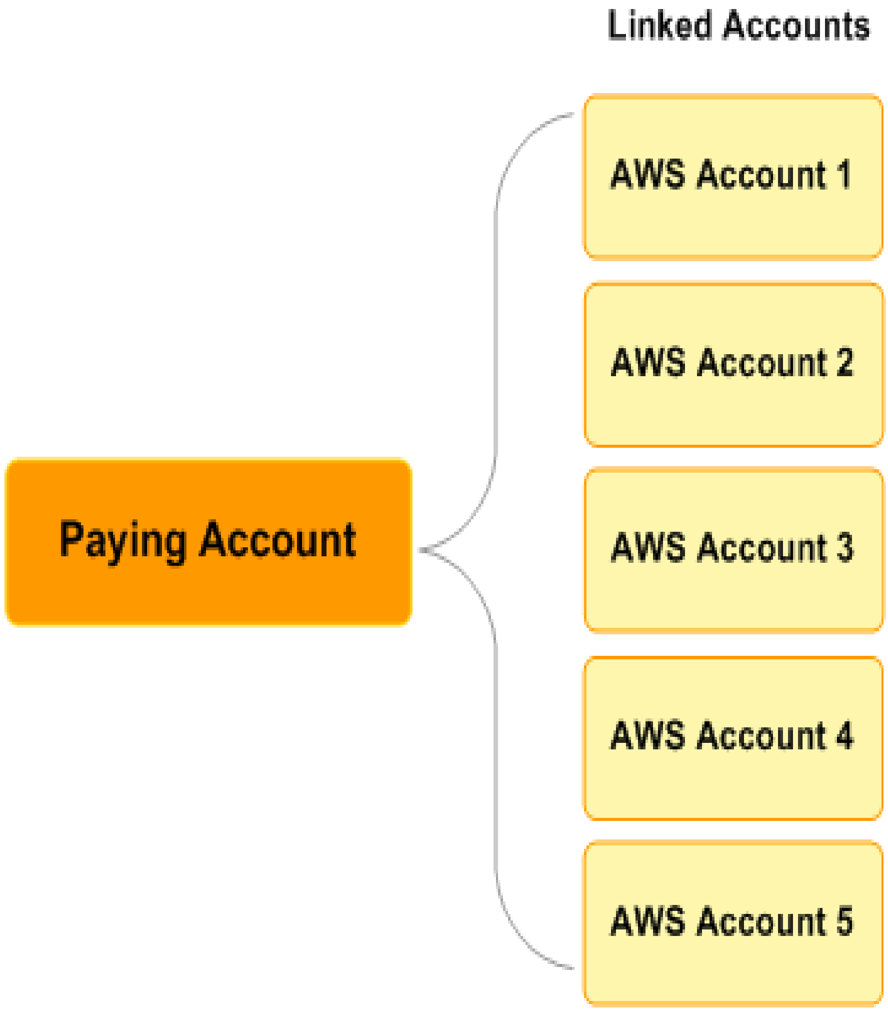
Every Company has….



Every Application has….

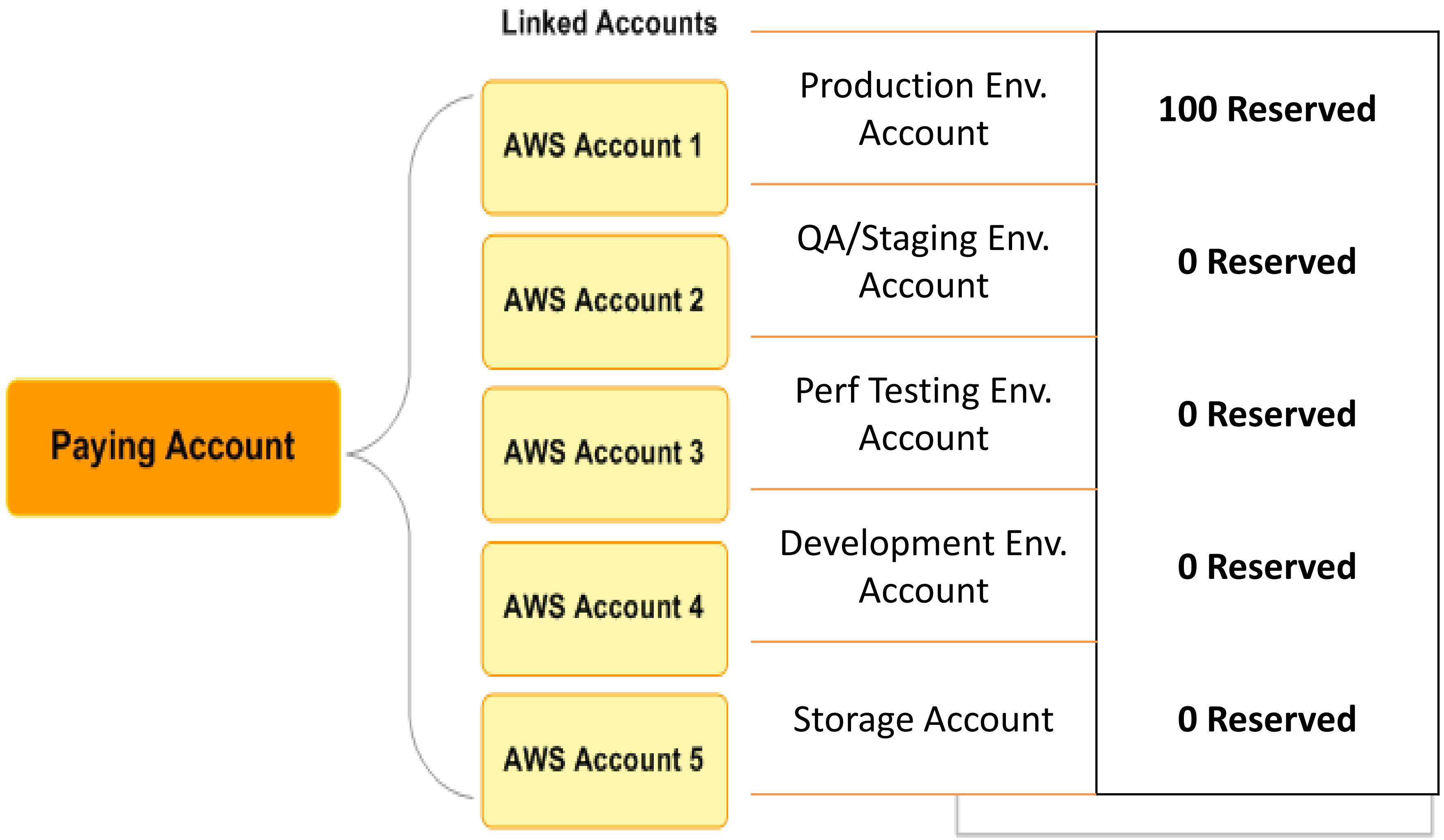


Consolidated Billing: Single payer for a group of accounts



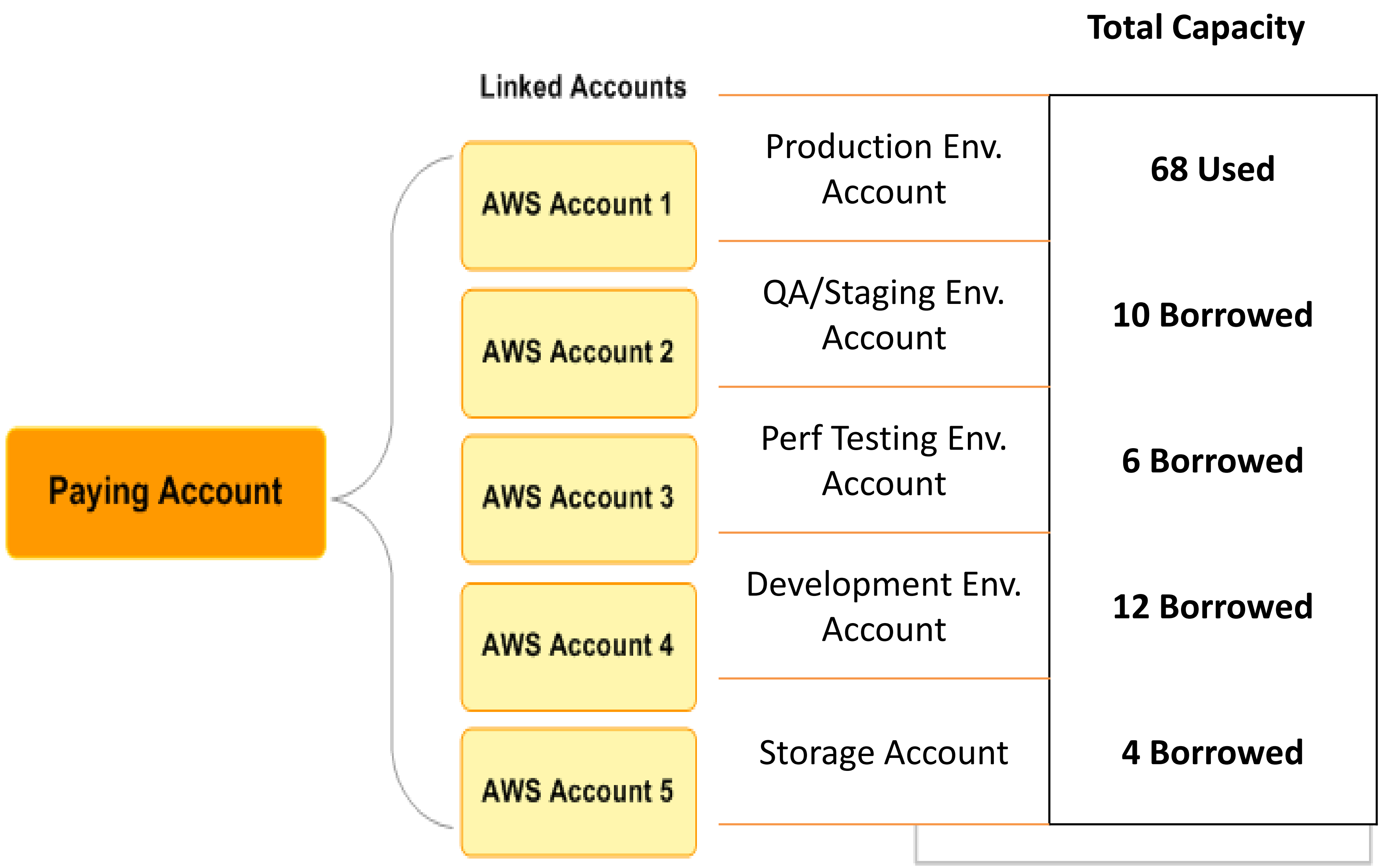
- **One Bill** for multiple accounts
- **Easy Tracking** of account charges (e.g., download CSV of cost data)
- **Volume Discounts** can be reached faster with combined usage
- **Reserved Instances** are shared across accounts (including RDS Reserved DBs)

Over-Reserve the Production Environment



	Total Capacity
Production Env. Account	100 Reserved
QA/Staging Env. Account	0 Reserved
Perf Testing Env. Account	0 Reserved
Development Env. Account	0 Reserved
Storage Account	0 Reserved

Consolidated Billing Borrows Unused Reservations



Consolidated Billing Advantages

- Production account is guaranteed to get burst capacity
  - Reservation is higher than normal usage level
  - Requests for more capacity always work up to reserved limit
  - Higher availability for handling unexpected peak demands
- No additional cost
  - Other lower priority accounts soak up unused reservations
  - Totals roll up in the monthly billing cycle

Building Cost-Aware Cloud Architectures

#1 Business Agility by Rapid Experimentation = Increased Revenue

#2 Business-driven Auto Scaling Architectures = Savings

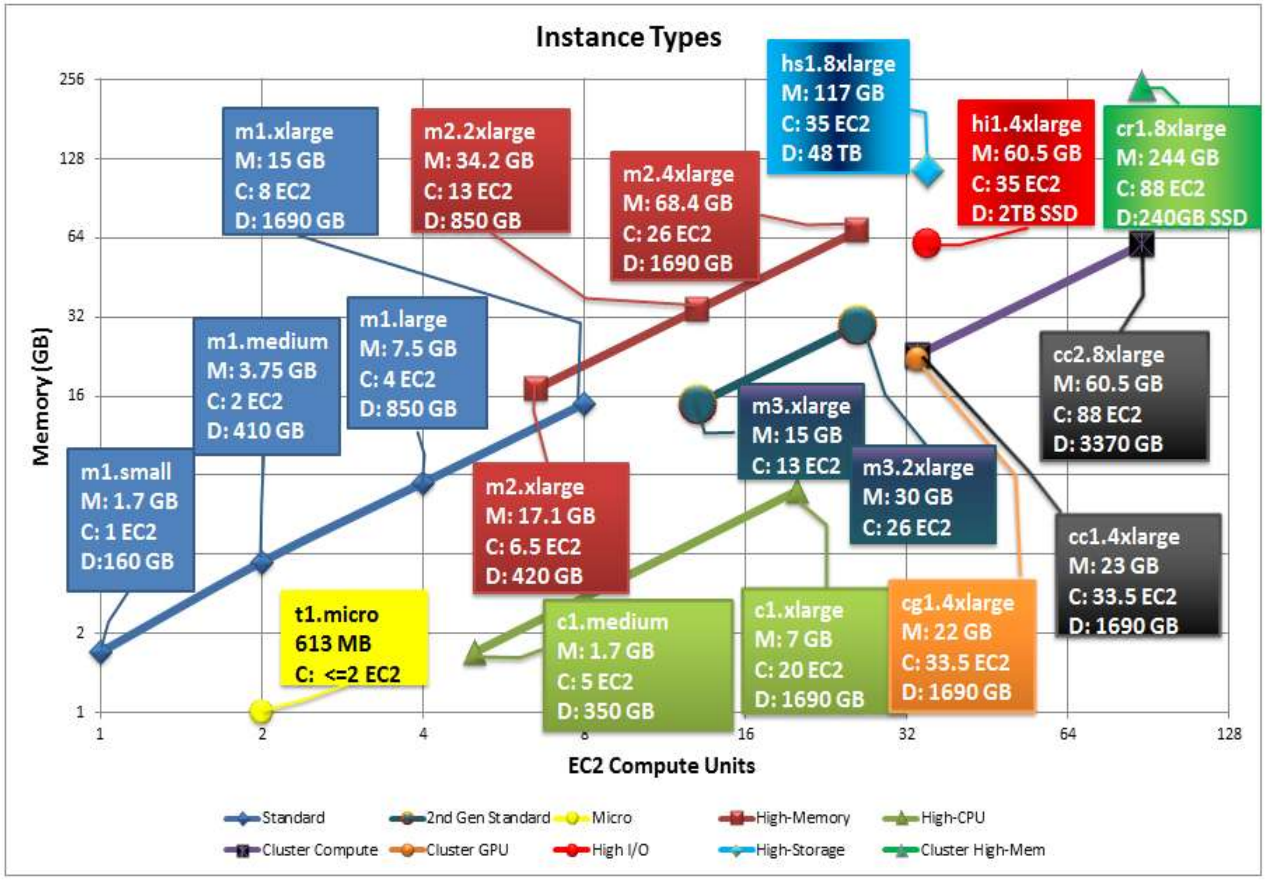
#3 Mix and Match Reserved Instances with On-Demand = Savings

#4 Consolidated Billing and Shared Reservations = Savings

Continuous optimization in your architecture results in *recurring savings* as early as your next month’s bill

Right-size your cloud: Use only what you need

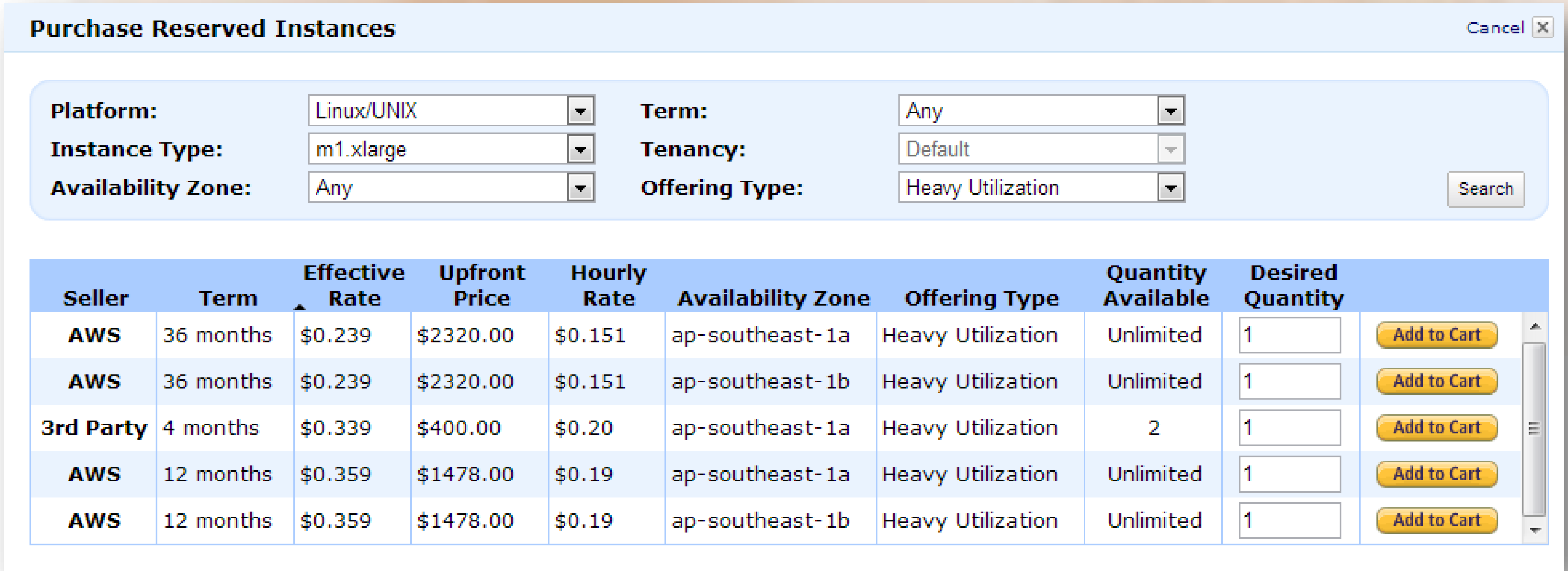
- An instance type for every purpose
- Assess your memory & CPU requirements
  - ~~Fit your application to the resource~~
  - Fit the resource to your application
- Only use a larger instance when needed



Reserved Instance Marketplace

Buy a smaller term instance
Buy instance with different OS or type
Buy a Reserved instance in different region

Sell your unused Reserved Instance
Sell unwanted or over-bought capacity
Further reduce costs by optimizing



Purchase Reserved Instances

Platform:	Linux/UNIX	Term:	Any
Instance Type:	m1.xlarge	Tenancy:	Default
Availability Zone:	Any	Offering Type:	Heavy Utilization

Seller	Term	Effective Rate	Upfront Price	Hourly Rate	Availability Zone	Offering Type	Quantity Available	Desired Quantity	
AWS	36 months	$0.239	$2320.00	$0.151	ap-southeast-1a	Heavy Utilization	Unlimited	1	Add to Cart
AWS	36 months	$0.239	$2320.00	$0.151	ap-southeast-1b	Heavy Utilization	Unlimited	1	Add to Cart
3rd Party	4 months	$0.339	$400.00	$0.20	ap-southeast-1a	Heavy Utilization	2	1	Add to Cart
AWS	12 months	$0.359	$1478.00	$0.19	ap-southeast-1a	Heavy Utilization	Unlimited	1	Add to Cart
AWS	12 months	$0.359	$1478.00	$0.19	ap-southeast-1b	Heavy Utilization	Unlimited	1	Add to Cart

Instance Type Optimization

Older m1 and m2 families

- Slower CPUs
- Higher response times
- Smaller caches (6MB)
- Oldest m1.xl 15GB/8ECU/48c
- Old m2.xl 17GB/6.5ECU/41c
- ~16 ECU/$/hr

Latest m3 family

- Faster CPUs
- Lower response times
- Bigger caches (20MB)
- Even faster for Java vs. ECU
- New m3.xl 15GB/13 ECU/50c
- 26 ECU/$/hr – 62% better!
- Java measured even higher
- Deploy fewer instances

Building Cost-Aware Cloud Architectures

#1 Business Agility by Rapid Experimentation = Increased Revenue

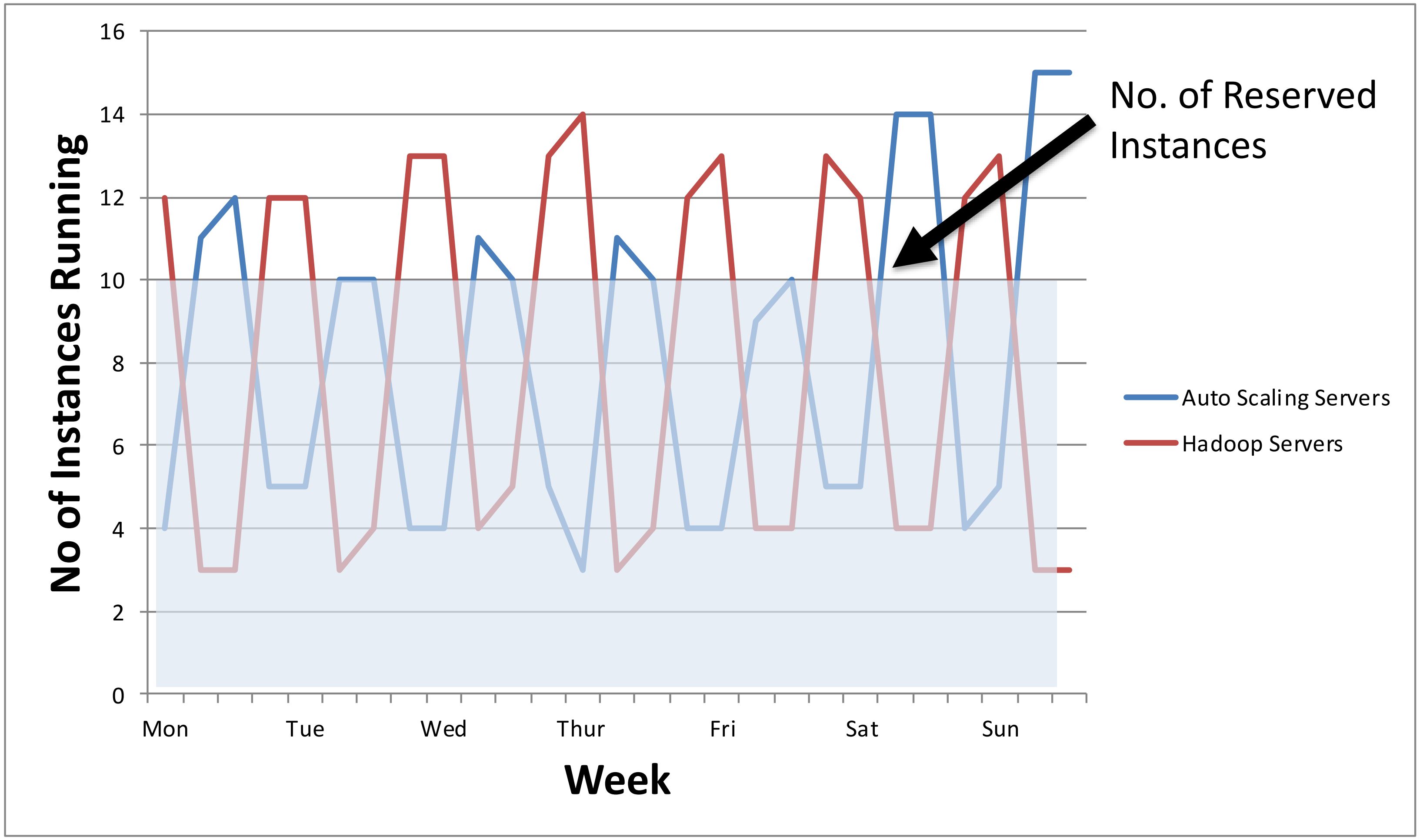
#2 Business-driven Auto Scaling Architectures = Savings

#3 Mix and Match Reserved Instances with On-Demand = Savings

#4 Consolidated Billing and Shared Reservations = Savings

#5 Always-on Instance Type Optimization = Recurring Savings

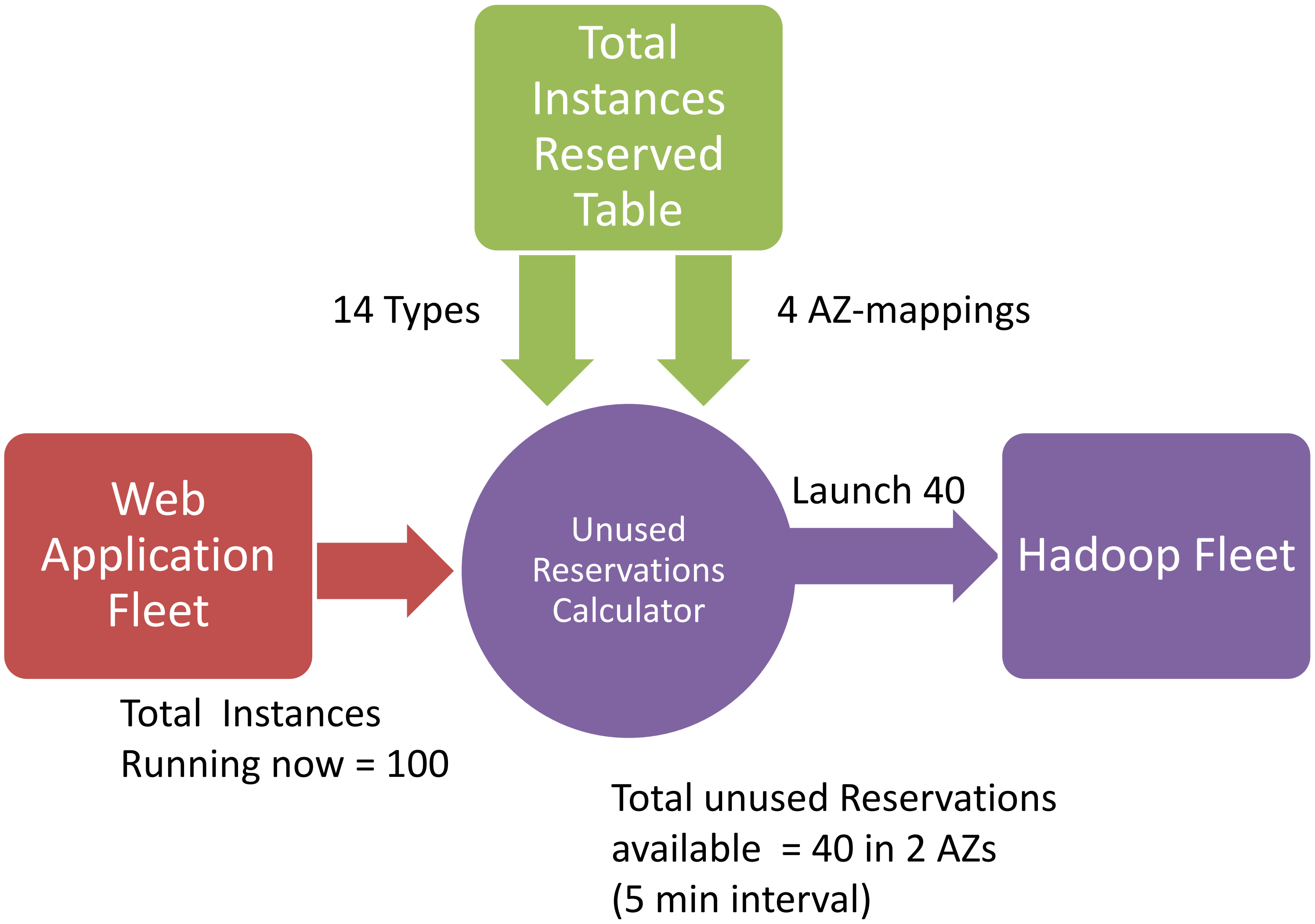
Follow the Customer (Run web servers) during the day



Follow the Money (Run Hadoop clusters) at night

Follow Your Customers

MONEY AV
SAVINGS ST



Total Instances Reserved Table
14 Types
4 AZ-mappings
Web Application Fleet
Unused Reservations Calculator
Launch 40
Hadoop Fleet
Total Instances Running now = 100
Total unused Reservations available = 40 in 2 AZs (5 min interval)

Soaking up unused reservations

Unused reserved instances is published as a metric

Netflix Data Science ETL Workload

- Daily business metrics roll-up
- Starts after midnight
- EMR clusters started using hundreds of instances

Netflix Movie Encoding Workload

- Long queue of high and low priority encoding jobs
- Can soak up 1000’s of additional unused instances

Building Cost-Aware Cloud Architectures

#1 Business Agility by Rapid Experimentation = Increased Revenue

#2 Business-driven Auto Scaling Architectures = Savings

#3 Mix and Match Reserved Instances with On-Demand = Savings

#4 Consolidated Billing and Shared Reservations = Savings

#5 Always-on Instance Type Optimization = Recurring Savings

#6 Follow the Customer (Run web servers) during the day
Follow the Money (Run Hadoop clusters) at night

Thank you!

Jinesh Varia and Adrian Cockcroft
jvaria@amazon.com @jinman
acockcroft@netflix.com @adrianco

Slideshare.net/Netflix Details

- Meetup S1E3 July – Featuring Contributors Eucalyptus, IBM, Paypal, Riot Games
  - http://techblog.netflix.com/2013/07/netflixoss-meetup-series-1-episode-3.html
- Lightning Talks March S1E2
  - http://www.slideshare.net/RuslanMeshenberg/netflixoss-meetup-lightning-talks-and-roadmap
- 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
- Security Architecture
  - http://www.slideshare.net/jason_chan/resilience-and-security-scale-lessons-learned/

Takeaways

Cloud Native Manages Scale and Complexity at Speed

NetflixOSS makes it easier for everyone to become Cloud Native

Rethink deployments and turn things off to save money!

http://netflix.github.com

http://techblog.netflix.com

http://slideshare.net/Netflix

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

@adrianco #netflixcloud @NetflixOSS