

# **Replacing Datacenter Oracle with Global Apache Cassandra on AWS**

July 11, 2011

Adrian Cockcroft

@adrianco #netflixcloud

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



# Netflix Inc.

*With more than 23 million subscribers in the United States and Canada, Netflix, Inc. is the world's leading Internet subscription service for enjoying movies and TV shows.*

## International Expansion

*We plan to expand into an additional market in the second half of 2011... If the second market meets our expectations... we will continue to invest and expand aggressively in 2012.*

Source: <http://ir.netflix.com>



# Building a Global Netflix Service

Netflix Cloud Migration  
Data Migration to Cassandra  
Highly Available and Globally Distributed Data  
Backups and Archives in the Cloud  
Monitoring Cassandra  
Contributions and Organization



# Why Use Public Cloud?



# Frictionless Deployment (JFDI)



# Get stuck with wrong config

Wait Wait File tickets

Ask permission Wait Wait

Wait Things We Don't Do Wait

Run out of space/power

Plan capacity in advance

Have meetings with IT Wait



# Better Business Agility





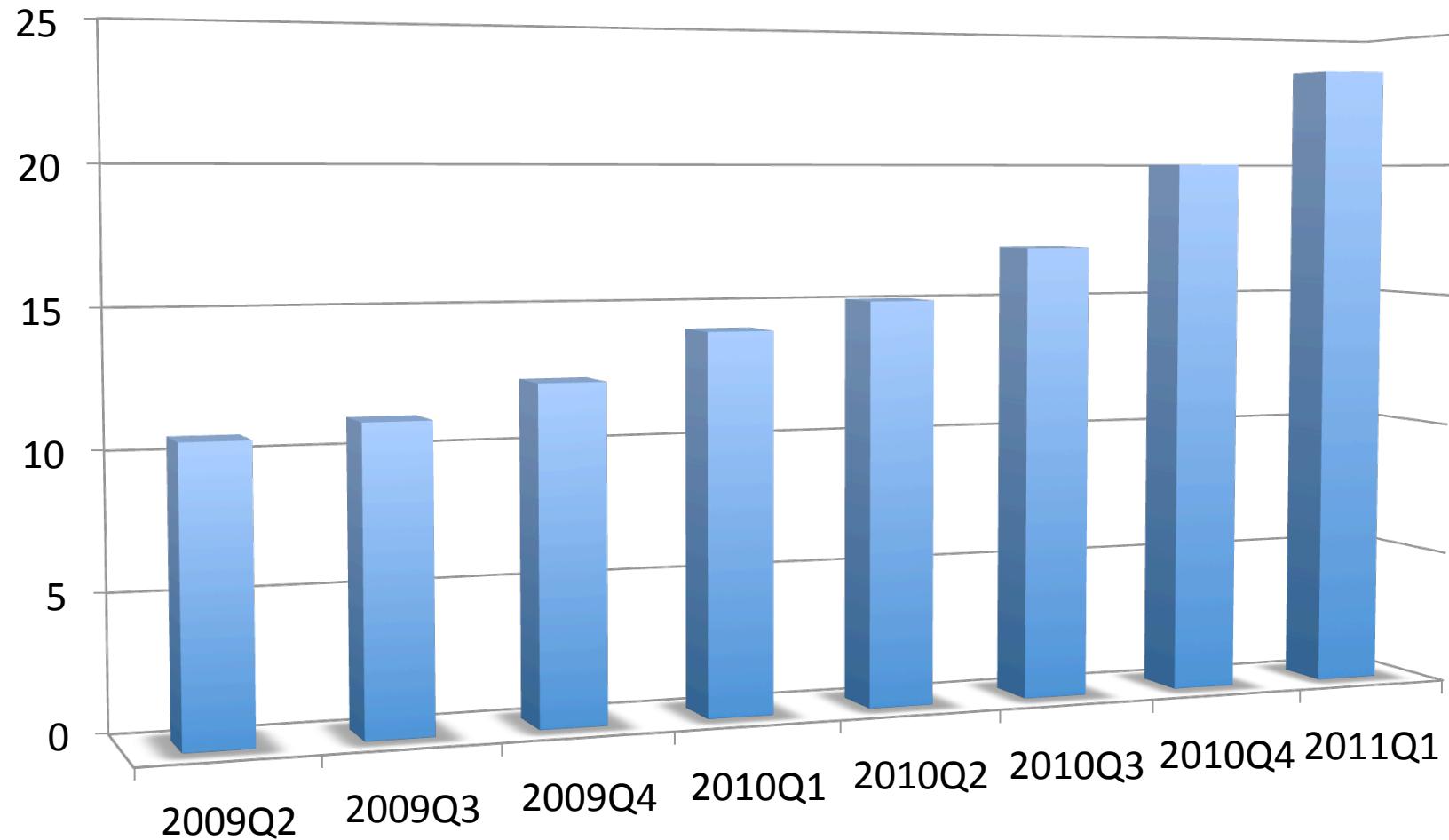
Netflix could not  
build new  
datacenters fast  
enough

Capacity growth is accelerating, unpredictable  
Product launch spikes - iPhone, Wii, PS3, XBox



# 23 Million Customers

2011-Q1 year/year customers +69%

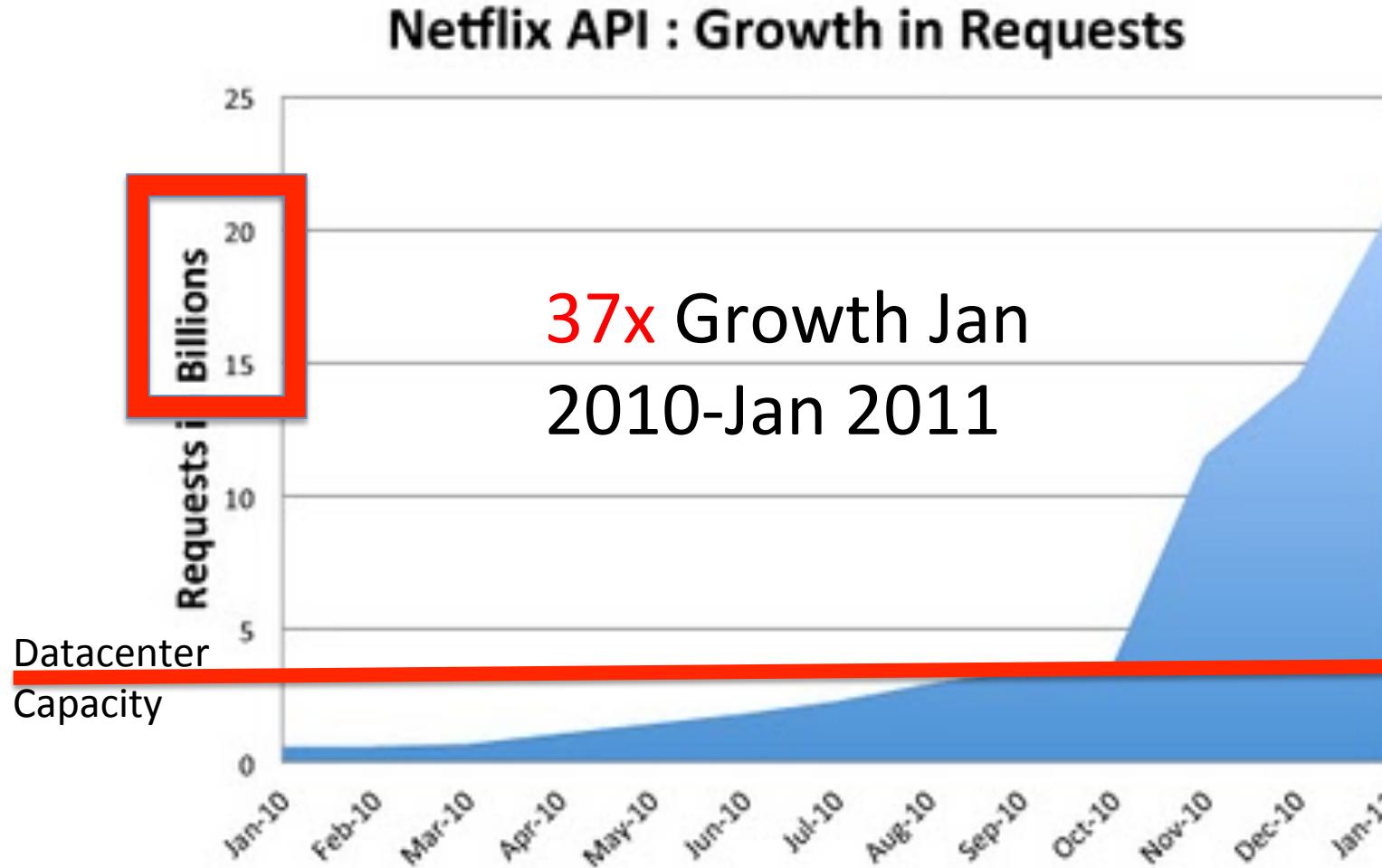


Source: <http://ir.netflix.com>



# Out-Growing Data Center

<http://techblog.netflix.com/2011/02/redesigning-netflix-api.html>



# Netflix.com is now ~100% Cloud

Account sign-up is currently being moved to cloud

All international product is cloud based

USA specific logistics remains in the Datacenter



**Netflix Choice was AWS with our  
own platform and tools**

Unique platform requirements and  
extreme agility and flexibility



# Leverage AWS Scale “the biggest public cloud”

AWS investment in features and automation  
Use AWS zones and regions for high availability,  
scalability and global deployment

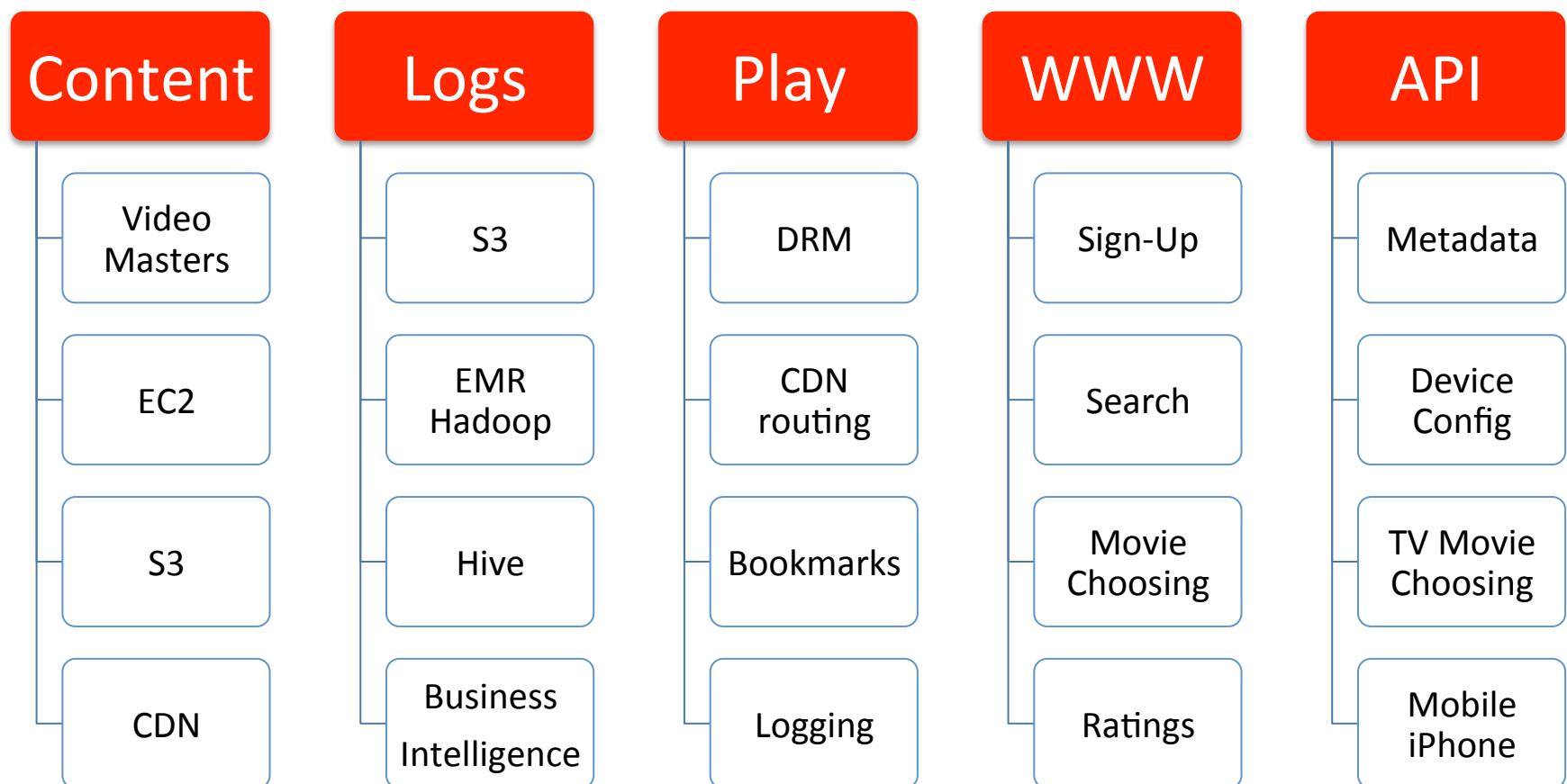


We want to use clouds,  
we don't have time to build them

Public cloud for agility and scale  
AWS because they are big enough to allocate thousands  
of instances per hour when we need to



# Netflix Deployed on AWS



# Port to Cloud Architecture

Short term investment, long term payback!

Pay down technical debt

Robust patterns



# Transition

- The Goals
  - Faster, Scalable, Available and Productive
- Anti-patterns and Cloud Architecture
  - The things we wanted to change and why
- Data Migration
  - Minimizing datacenter dependencies

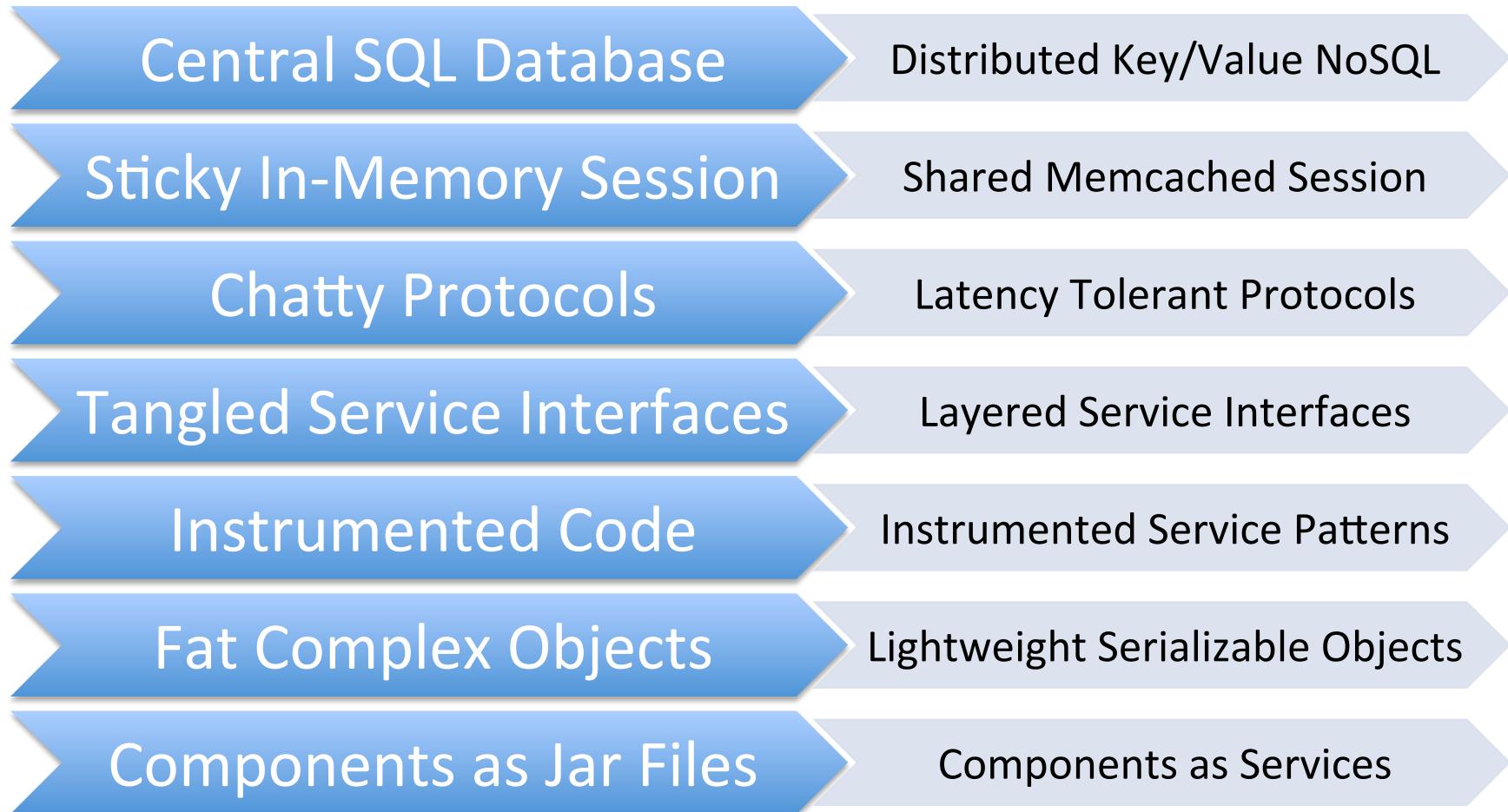


# Datacenter Anti-Patterns

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



# Old Datacenter vs. New Cloud Arch



# The Central SQL Database

- Datacenter has central Oracle databases
  - Everything in one place is convenient until it fails
  - Customers, movies, history, configuration
- Schema changes require downtime

*Anti-pattern impacts scalability, availability*



# The Distributed Key-Value Store

- Cloud has many key-value data stores
  - More complex to keep track of, do backups etc.
  - Each store is much simpler to administer
  - Joins take place in java code
- No schema to change, no scheduled downtime
- Latency for typical queries
  - Memcached is dominated by network latency <1ms
  - Cassandra replication takes a few milliseconds
  - Oracle for simple queries is a few milliseconds
  - SimpleDB replication and REST auth overheads >10ms



# Data Migration to Cassandra

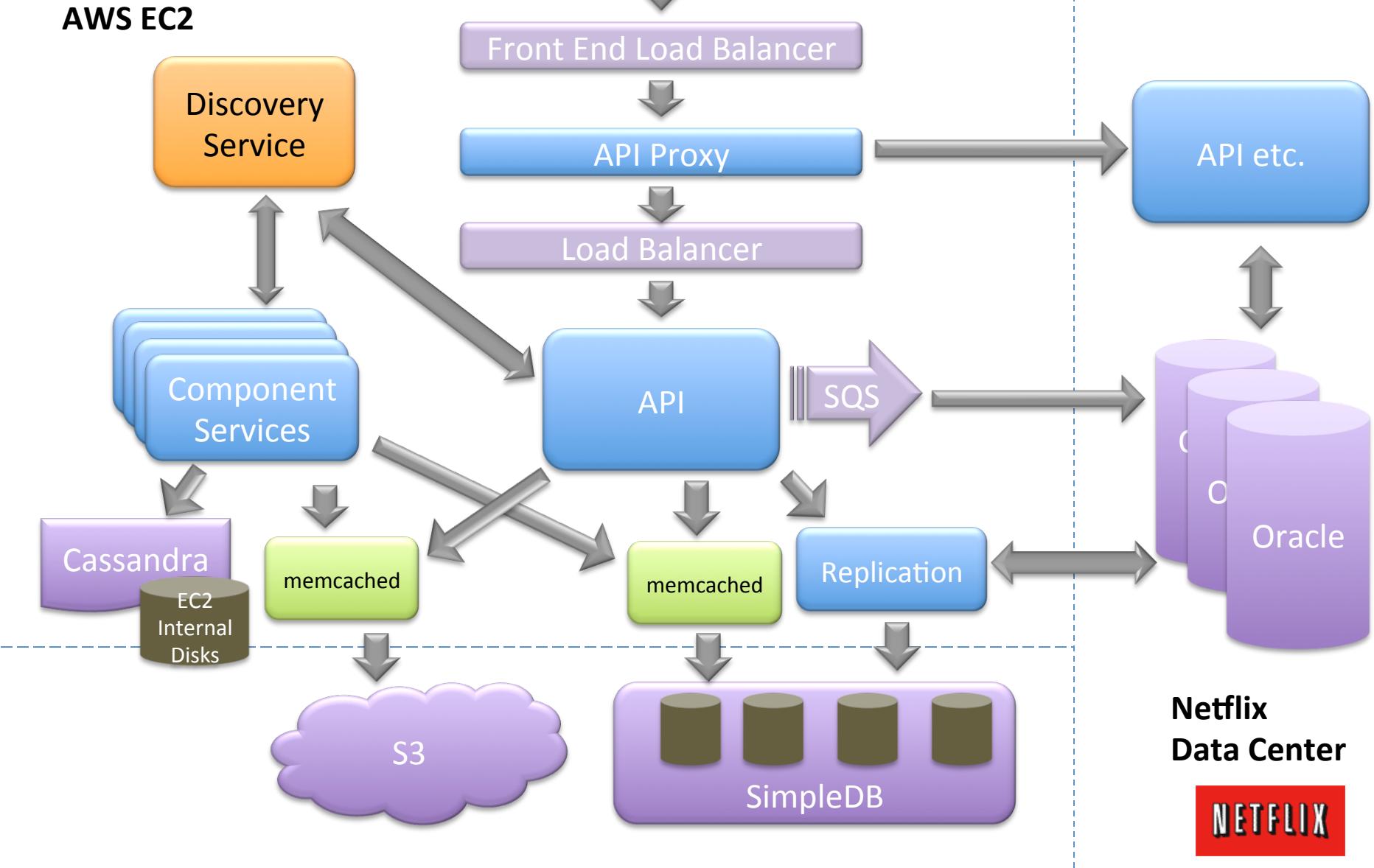


# Transitional Steps

- Bidirectional Replication
  - Oracle to SimpleDB
  - Queued reverse path using SQS
  - Backups remain in Datacenter via Oracle
- New Cloud-Only Data Sources
  - Cassandra based
  - No replication to Datacenter
  - Backups performed in the cloud



# API



# Cutting the Umbilical

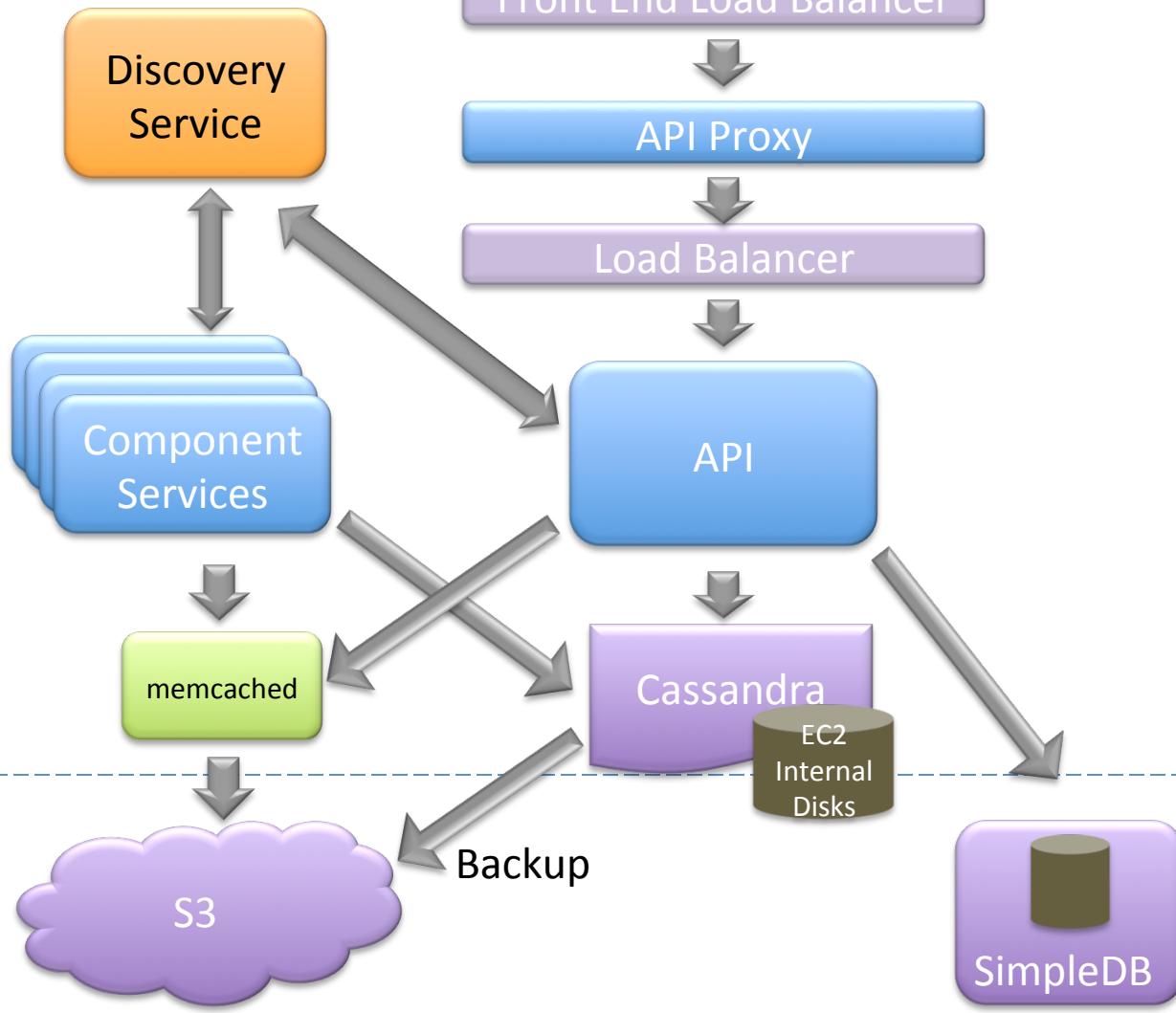
- Transition Oracle Data Sources to Cassandra
  - Offload Datacenter Oracle hardware
  - Free up capacity for growth of remaining services
- Transition SimpleDB+Memcached to Cassandra
  - Primary data sources that need backup
  - Keep simple use cases like configuration service
- New challenges
  - Backup, restore, archive, business continuity
  - Business Intelligence integration



# API



AWS EC2



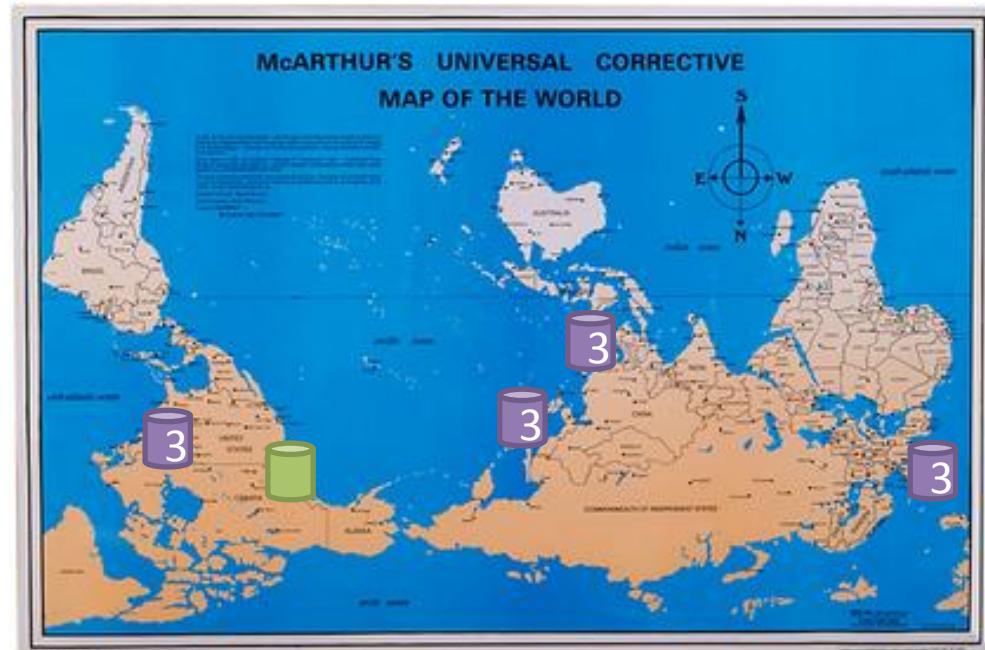
# High Availability

- Cassandra stores 3 local copies, 1 per zone
  - Synchronous access, durable, highly available
  - Read/Write One fastest, least consistent - ~1ms
  - Read/Write Quorum 2 of 3, consistent - ~3ms
- AWS Availability Zones
  - Separate buildings
  - Separate power etc.
  - Close together



# Remote Copies

- Cassandra duplicates across AWS regions
  - Asynchronous write, replicates at destination
  - Doesn't directly affect local read/write latency
- Global Coverage
  - Business agility
  - Follow AWS...
- Local Access
  - Better latency
  - Fault Isolation



NETFLIX

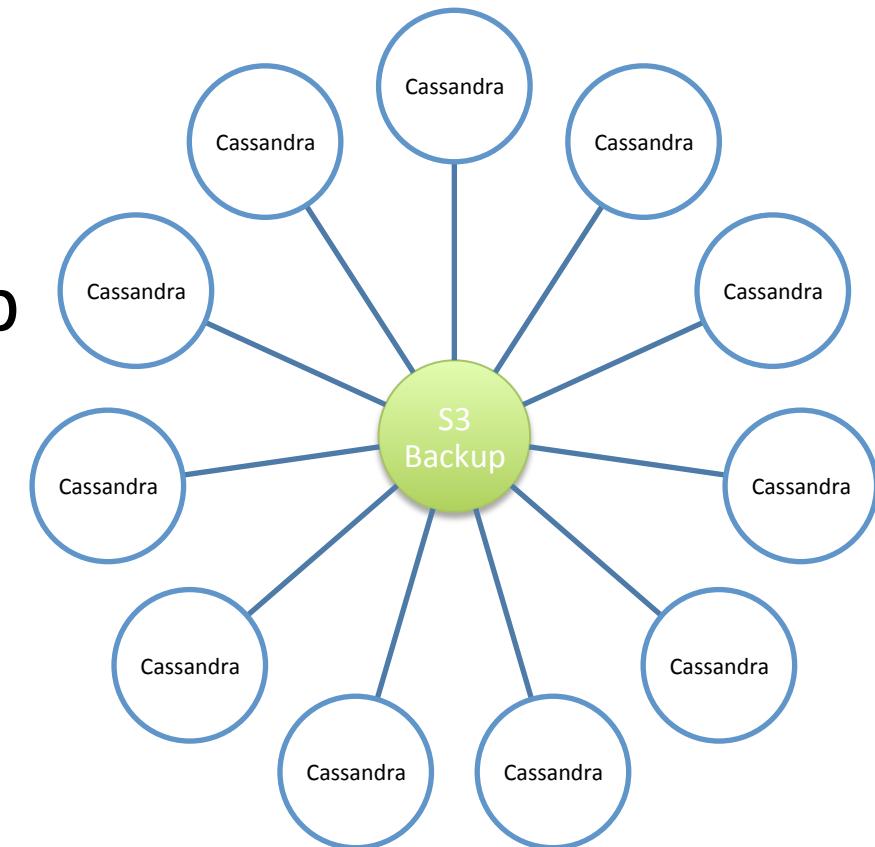
# Cassandra Backup

- Full Backup
  - Cron on each node
  - Snapshot -> tar.gz -> S3
- Incremental
  - SSTable write triggers copy to S3
- Continuous
  - Scrape commit log
  - Write to EBS every 30s



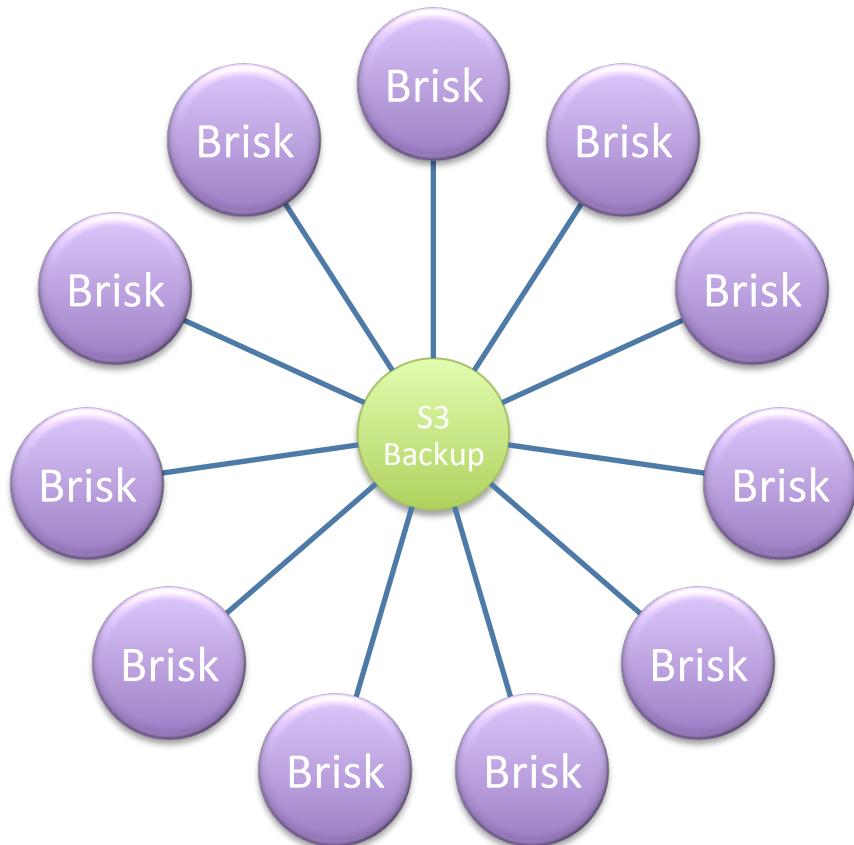
# Cassandra Restore

- Full Restore
  - Replace previous data
- New Ring from Backup
  - New name old data
  - One line command!



# Cassandra Data Extraction

- Business Intelligence
  - Re-normalize data using Hadoop job
- Daily Extraction
  - Create Brisk ring
  - Extract backup
  - Run Hadoop job
  - Remove Brisk ring
  - Under 1hr...



# Cassandra Online BI

- Intra-Day Extraction
  - Use split Brisk ring
  - Size each separately
  - Hourly Hadoop job



# Cassandra Archive

Appropriate level of paranoia needed...

- Archive could be un-readable
  - Base on restored S3 backup and BI extracted data
- Archive could be stolen
  - Encrypt archive
- AWS East Region could have a problem
  - Copy data to AWS West
- Production AWS Account could have an issue
  - Separate Archive account with no-delete S3 ACL
- AWS S3 could have a global problem
  - Create an extra copy on a different cloud vendor



# Tools and Automation

- Developer and Build Tools
  - Jira, Perforce, Eclipse, Jenkins, Ivy, Artifactory
  - Builds, creates .war file, .rpm, bakes AMI and launches
- Custom Netflix Application Console
  - AWS Features at Enterprise Scale (hide the AWS security keys!)
  - Auto Scaler Group is unit of deployment to production
- Open Source + Support
  - Apache, Tomcat, Cassandra, Hadoop, OpenJDK, CentOS
  - Datastax support for Cassandra, AWS support for Hadoop via EMR
- Monitoring Tools
  - Datastax Opscenter for monitoring Cassandra
  - AppDynamics – Developer focus for cloud <http://appdynamics.com>



# Developer Migration

- Detailed SQL to NoSQL Transition Advice
  - Sid Anand - QConSF Nov 5<sup>th</sup> – Netflix' Transition to High Availability Storage Systems
  - Blog - <http://practicalcloudcomputing.com/>
  - Download Paper PDF - <http://bit.ly/bhOTLu>
- Mark Atwood, "Guide to NoSQL, redux"
  - YouTube <http://youtu.be/zAbFRiyT3LU>



# Cloud Operations

Cassandra Use Cases  
Model Driven Architecture  
Capacity Planning & Monitoring  
Chaos Monkey



# Cassandra Use Cases

- Key by Customer
  - Several separate Cassandra rings, read-intensive
  - Sized to fit in memory using m2.4xl Instances
- Key by Customer:Movie – e.g. Viewing History
  - Growing fast, write intensive – m1.xl instances
  - Sized to hold hot data in memory only
- Large scale data logging – lots of writes
  - Column data expires after time period
  - Working on using distributed counters...



# Model Driven Architecture

- Datacenter Practices
  - Lots of unique hand-tweaked systems
  - Hard to enforce patterns
- Model Driven Cloud Architecture
  - Perforce/Ivy/Jenkins based builds for *everything*
  - Every production instance is a pre-baked AMI
  - Every application is managed by an Autoscaler

*Every change is a new AMI*



# Netflix Platform Cassandra AMI

- Tomcat server
  - Always running, registers with platform
  - Manages Cassandra state, tokens, backups
- SimpleDB configuration
  - Stores token slots and options
  - Avoids circular “bootstrap problems”
- Removed Root Disk Dependency on EBS
  - Use S3 backed AMI for stateful services
  - Normally use EBS backed AMI for fast provisioning



# Netflix App Console

← → C nactest.netflix.com/application/show/cass\_perf\_sr  

 NETFLIX Application Console (test) Region: us-east-1 (v)  CMC:

Home Apps Images Auto Scaling Load Balancers Instances EBS RDS Tasks

### Application Details

[Edit Application](#) [Delete Application](#) [Edit Application Security Access](#)

Name: cass\_perf\_sr  
Warning: Punctuation in name prevents use as frontend service.

Type: Web Service  
Description: Single region performance test  
Owner: Adrian  
Email: acockcroft@netflix.com  
Create Time: 2011-06-13 14:04:45 PDT  
Update Time: 2011-06-13 14:04:45 PDT

### Pattern Matches

Auto Scaling: [cass\\_perf\\_sr--useast1](#) [cass\\_perf\\_sr--useast1d](#) [cass\\_perf\\_sr--useast1a](#)

Load Balancers: [cass\\_perf\\_sr](#)

Security Groups: [cass\\_perf\\_sr](#)

Launch Configurations: [cass\\_perf\\_sr](#)

Running Instances: [Running Instance List](#)



# Auto Scale Group Configuration

NETFLIX Application Console (test)      Region: us-east-1 (v)      CMC:

Home Apps Images Auto Scaling Load Balancers Instances EBS RDS Tasks

### Auto Scaling Group Details

[Edit Auto Scaling Group](#) [Delete Auto Scaling Group](#) [Create new Launch Config](#) [Prepare Rolling Push](#)

[Manage Cluster of Sequential ASGs](#)

Name:	cass_perf_sr--useast1d
Launch Configuration:	<a href="#">cass_perf_sr--useast1d-201106131415</a>
Application:	<a href="#">cass_perf_sr</a>
Detail:	useast1d
Min Instances:	4
Desired Instances:	4
Max Instances:	4
Cool Down:	10 seconds
ASG Health Check Type:	EC2 (Replace terminated instances)
ASG Health Check Grace Period:	600 seconds
Availability Zones:	[us-east-1d]
AZ Rebalancing:	Enabled
New Instance Launching:	Enabled
Created Time:	2011-06-13 14:15:29 PDT
Load Balancers:	-
Activities:	<p>At 2011-06-13T21:15:29Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 4. At 2011-06-13T21:15:37Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 4. : Launching a new EC2 instance: i-01addb6f (100% done) (Status: Successful)</p> <p>At 2011-06-13T21:15:29Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 4. At 2011-06-13T21:15:37Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 4. : Launching a new EC2 instance: i-05addb6b (100% done) (Status: Successful)</p>



# Chaos Monkey



- Make sure systems are resilient
  - Allow any instance to fail without customer impact
- Chaos Monkey hours
  - Monday-Thursday 9am-3pm random instance kill
- Application configuration option
  - Apps now have to opt-out from Chaos Monkey
- Computers (Datacenter or AWS) randomly die
  - Fact of life, but too infrequent to test resiliency



# Capacity Planning & Monitoring



# Capacity Planning in Clouds

(a few things have changed...)

- Capacity is expensive
- Capacity takes time to buy and provision
- Capacity only increases, can't be shrunk easily
- Capacity comes in big chunks, paid up front
- Planning errors can cause big problems
- Systems are clearly defined assets
- Systems can be instrumented in detail
- Depreciate assets over 3 years (reservations!)



# Data Sources

## External Testing

- External URL availability and latency alerts and reports – Keynote
- Stress testing - SOASTA

## Request Trace Logging

- Netflix REST calls – Chukwa to DataOven with GUID transaction identifier
- Generic HTTP – AppDynamics service tier aggregation, end to end tracking

## Application logging

- Tracers and counters – log4j, tracer central, Chukwa to DataOven
- Trackid and Audit/Debug logging – DataOven, Appdynamics GUID cross reference

## JMX Metrics

- Application specific real time – Datastax Opscenter, Appdynamics
- Service and SLA percentiles – Appdynamics, Epic logged to DataOven

## Tomcat and Apache logs

- Stdout logs – S3 – DataOven
- Standard format Access and Error logs – S3 – DataOven

## JVM

- Garbage Collection – Appdynamics
- Memory usage, call stacks, resource/call - AppDynamics

## Linux

- system CPU/Net/RAM/Disk metrics – AppDynamics
- SNMP metrics – Epic, Network flows – boundary.com

## AWS

- Load balancer traffic – Amazon Cloudwatch, SimpleDB usage stats
- System configuration - CPU count/speed and RAM size, overall usage - AWS



# AppDynamics

How to look deep inside your cloud applications

- Automatic Monitoring
  - Base AMI bakes in all monitoring tools
  - Outbound calls only – no discovery/polling issues
  - Inactive instances removed after a few days
- Incident Alarms (deviation from baseline)
  - Business Transaction latency and error rate
  - Alarm thresholds discover their own baseline
  - Email contains URL to Incident Workbench UI



# AppDynamics Monitoring of Cassandra – Automatic Discovery

**Request: a4c39b7f-c310-48ba-bca3-56bc7cf86ec6**

USER EXPERIENCE	EXECUTION TIME	TIMESTAMP	BUSINESS TRANSACTION	REQUEST GUID
VERY_SLOW	4801 ms	04/29/11 03:57:37 PM	/bible/words	a4c39b7f-c310-48ba-bca3-56bc7cf86ec6

**Request Flow Map**

The Request Flow Map shows a slow request (4666 ms, 97.2%) from the WEB layer to the CASSANDRA layer. A 'Drill Down' button is available for both layers.

**Call Drill Down (Request: a4c39b7f-c310-48ba-bca3-56bc7cf86ec6)**

**SUMMARY**

Execution Time: 4801 ms. Node JETTY. Timestamp: 04/29/11 03:57:28 PM.

Name	Time (ms)	External Calls	Details
Servlet - WordsServlet15Servlet - WordsServlet doGet	3 ms (self)	0.1 %	<a href="#">View Details</a>
com.appdynamics.bible.xrefs.cassandra.CassandraHelper getAllVersesWithWord:160	0 ms (self)	0 %	<a href="#">View Details</a>
<b>org.apache.cassandra.thrift.Cassandra\$Client:get_slice:512</b>	198 ms (self)	4.1 %	<a href="#">Custom</a> <a href="#">View Details</a>

**CALL GRAPH**

Set as Root Reset Root Callgraph navigation help

**HOT SPOTS**

**SQL CALLS**

**HTTP PARAMS**

**COOKIES**

**USER DATA**

**ERROR DETAILS**

**HARDWARE / JVM**

**ADDITIONAL DATA**

**CUSTOM Calls**

Calling Method: Cassandra\$Client.get\_slice

Row key: he  
Column family: Words  
Consistency level: QUORUM

[Drill Down into Call](#)

**Call Drill Down (Request: a4c39b7f-c310-48ba-bca3-56bc7cf86ec6)**

**SUMMARY**

Execution Time: 104 ms. Node CASSANDRA. Timestamp: 04/29/11 03:57:28 PM.

Name	Time (ms)	External Calls	Details
org.apache.cassandra.thrift.CassandraServer:get_slice	101 ms (self)	97.1 %	<a href="#">View Details</a>
org.apache.cassandra.thrift.CassandraServer:multigetSliceInternal:273	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.thrift.CassandraServer:getSlice:197	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.thrift.CassandraServer:readColumnFamily:100	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.service.StorageProxy:read:293	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.service.StorageProxy:fetchRows:390	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.service.ReadCallback:get:108	0 ms (self)	0 %	<a href="#">View Details</a>
org.apache.cassandra.utils.SimpleCondition:await:54	0 ms (self)	0 %	<a href="#">View Details</a>



# DataStax OpsCenter

The image displays three screenshots of the DataStax OpsCenter interface, highlighting its monitoring and management capabilities for Apache Cassandra clusters.

- Metrics View:** Shows a line graph of "Cluster Write Operations" over time (2:41PM - Feb 3 3:01PM). The Y-axis represents operations per second, ranging from 0/sec to 70000/sec. The X-axis shows time intervals from 2:41PM to 3:01PM. A legend indicates the "Total" metric. The graph shows a significant spike in write operations starting around 2:56PM, reaching approximately 70,000 ops/sec by 3:01PM. The interface includes a sidebar with navigation links like Dashboard, Events & Alerts, Cluster Management, Performance, Data Modeling, Data Explorer, Settings, and Logout.
- Cluster Management View:** Displays a ring diagram of the "MainCluster". Nodes are represented by colored circles (green, yellow, red) connected by lines forming a ring. Each node has a unique identifier (e.g., 0000000, 7777777, 6666666, etc.) displayed above it. A context menu is open over a green node labeled "node11.o.datastax.com (4CCCCCCC) Active". The menu options include: Actions (View Metrics, View Replication, Cleanup, Compact, Flush, Repair, Compression, Drain, Move), Add, and Balance Cluster.
- Datacenter View:** Shows a hierarchical view across three datacenters: Datacenter 1, Datacenter 2, and Datacenter 3. Each datacenter contains multiple nodes, each with a unique identifier. Dotted lines connect nodes between datacenters, indicating replication or consistency groups. The interface includes a sidebar with navigation links like Dashboard, Events & Alerts, Cluster Management, Performance, Data Modeling, Data Explorer, Keyspaces (Keyspace1, TempKeyspace, system), Settings, and Logout.



# Netflix Contributions to Cassandra

- Cassandra as a mutable toolkit
  - Cassandra is in Java, pluggable, well structured
  - Netflix has a building full of Java engineers....
- Actual Contributions delivered in 0.8
  - First prototype of off-heap row cache (Vijay)
  - Incremental backup SSTable write callback
- Work In Progress
  - AWS integration and backup using Tomcat helper
  - Total re-write of Hector Java client library (Eran)



# Netflix “NoOps” Organization

Marketing & Advertising Site  
for Customer Acquisition

Member Site Personalization  
for Customer Retention

Cloud Ops  
Reliability  
Engineering

Database  
Engineering

Build Tools  
and  
Automation

Platform  
Development

Cloud  
Performance

Cloud  
Solutions

Cassandra

Cassandra

Perforce  
Jenkins

Cassandra

Cassandra

Cassandra

AWS

AWS

AWS

AWS

AWS

AWS



# Takeaway

*Netflix is using Cassandra on AWS as a key infrastructure component of its globally distributed streaming product.*

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

@adrianco #netflixcloud



# 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 Availability Zones – US-East, US-West, EU-Eire, Asia-Singapore, Asia-Japan
- 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)
- SDB – Simple Data Base (hosted http based NoSQL data store)
- 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 (extension of enterprise datacenter network into cloud)
- IAM – Identity and Access Management (fine grain role based security keys)

