Global Netflix

Replacing Datacenter Oracle with Global Apache Cassandra on AWS

October 24th, 2011

Adrian Cockcroft

@adrianco #netflixcloud

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

Netflix Inc.

With over 25 million members in the United States, Canada and Latin America, Netflix, Inc. is the world's leading Internet subscription service for enjoying movies and TV shows.

International Expansion

Netflix, Inc., the leading global Internet movie subscription service, today announced it will expand to the United Kingdom and Ireland in early 2012.

Source: http://ir.netflix.com

NETFLIX

Building a Global Netflix Service

Netflix Cloud Migration

Highly Available and Globally Distributed Data

Scalability and Performance

Why Use Public Cloud?

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

NETFLIX

Better Business Agility

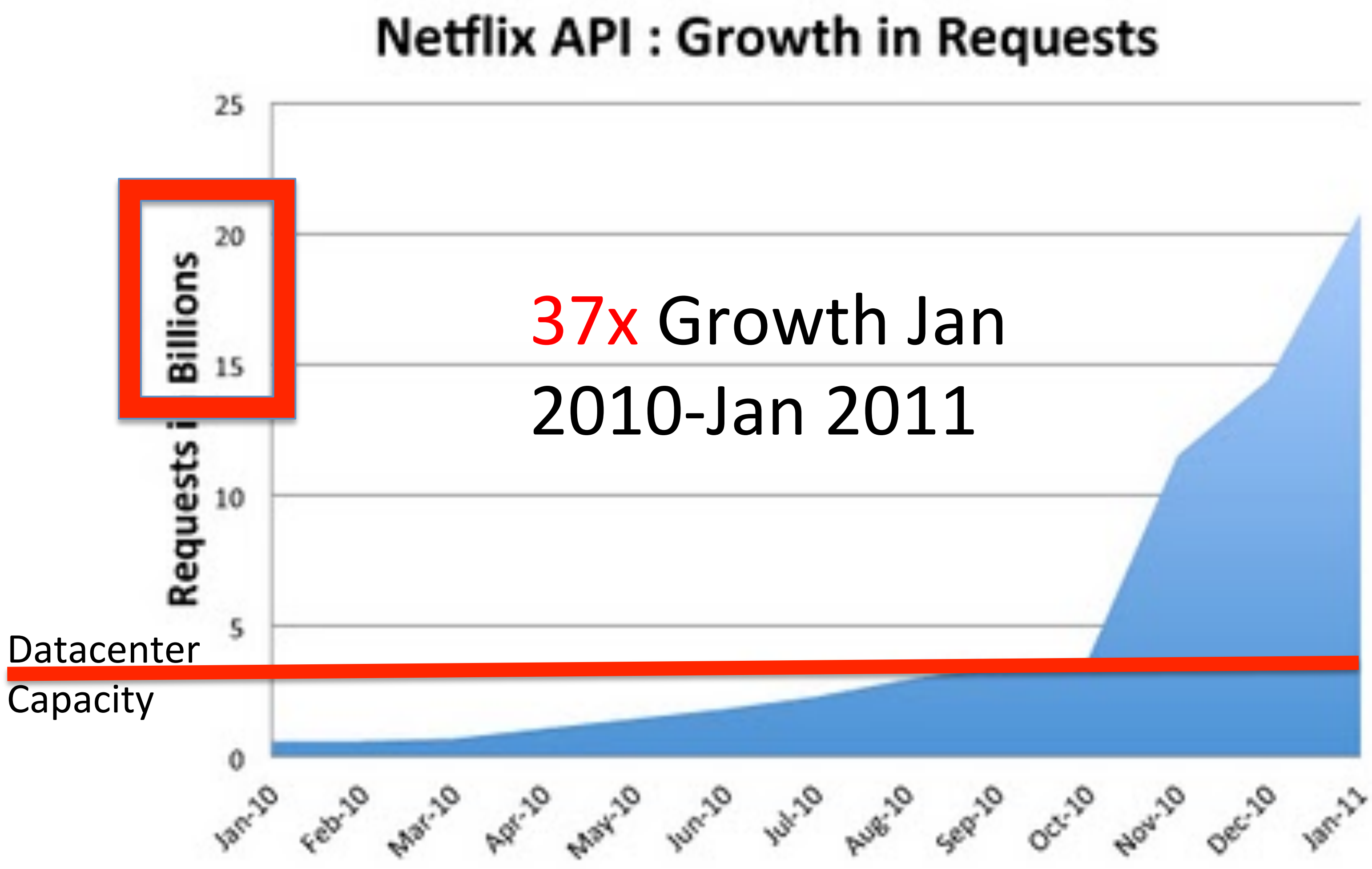
Netflix could not build new datacenters fast enough

Capacity growth is accelerating, unpredictable

Product launch spikes - iPhone, Wii, PS3, XBox

Out-Growing Data Center

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



37x Growth Jan 2010-Jan 2011

Datacenter Capacity

Netflix.com is now ~100% Cloud

A few small back end data sources still in progress

All international product is cloud based

USA specific logistics remains in the Datacenter

Working aggressively on billing, PCI compliance on AWS

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

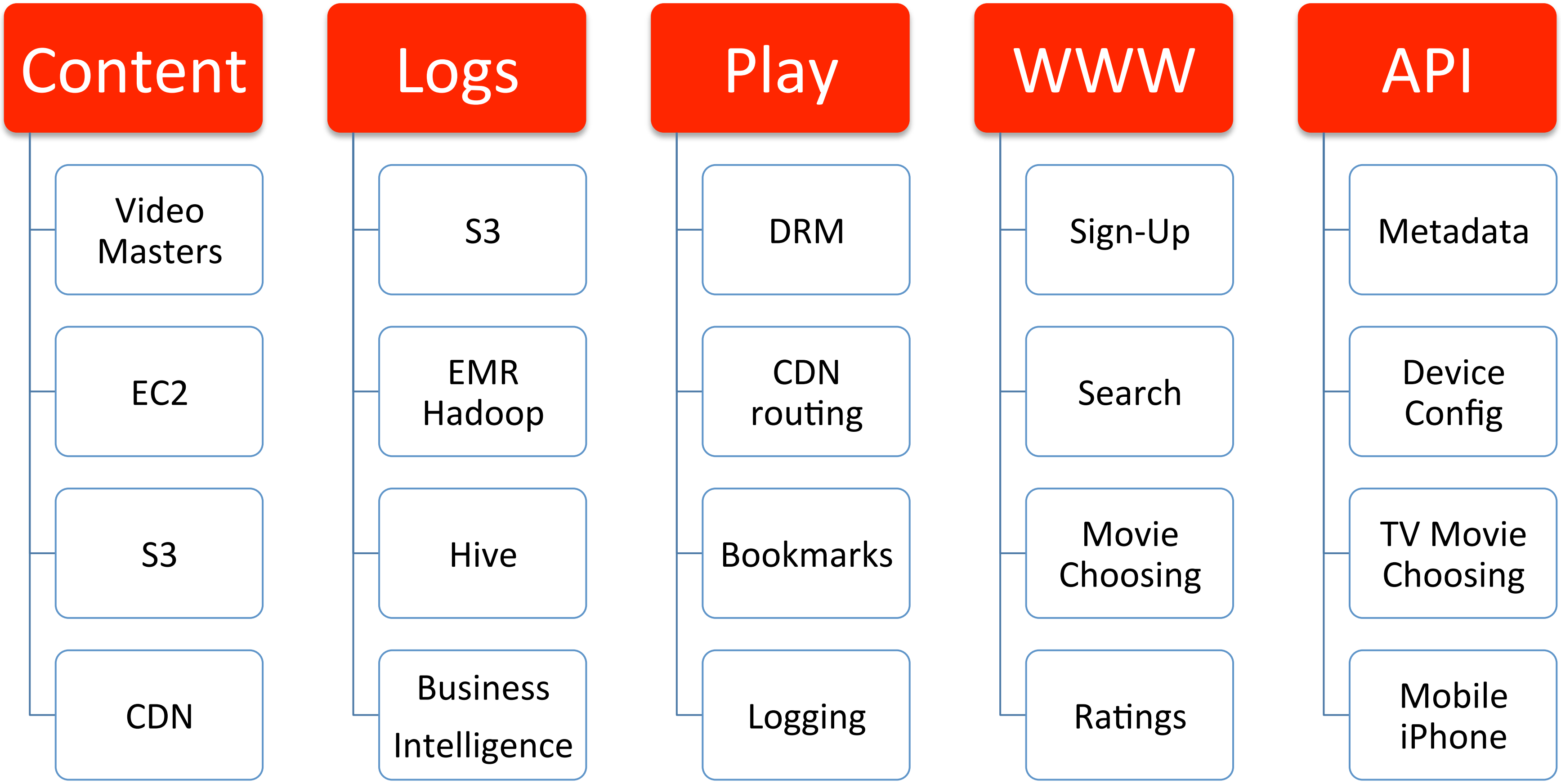
We want to use clouds, we don’t have time to build them

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

amazon webservices™



Content	Logs	Play	WWW	API
Video Masters	S3	DRM	Sign-Up	Metadata
EC2	EMR Hadoop	CDN routing	Search	Device Config
S3	Hive	Bookmarks	Movie Choosing	TV Movie Choosing
CDN	Business Intelligence	Logging	Ratings	Mobile iPhone

Akamai Limelight NETWORKS Level(3) COMMUNICATIONS NETFLIX

Datacenter Anti-Patterns

What did we do in the datacenter that prevented us from meeting our goals?

Old Datacenter vs. New Cloud Arch

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

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

🚫 DBA

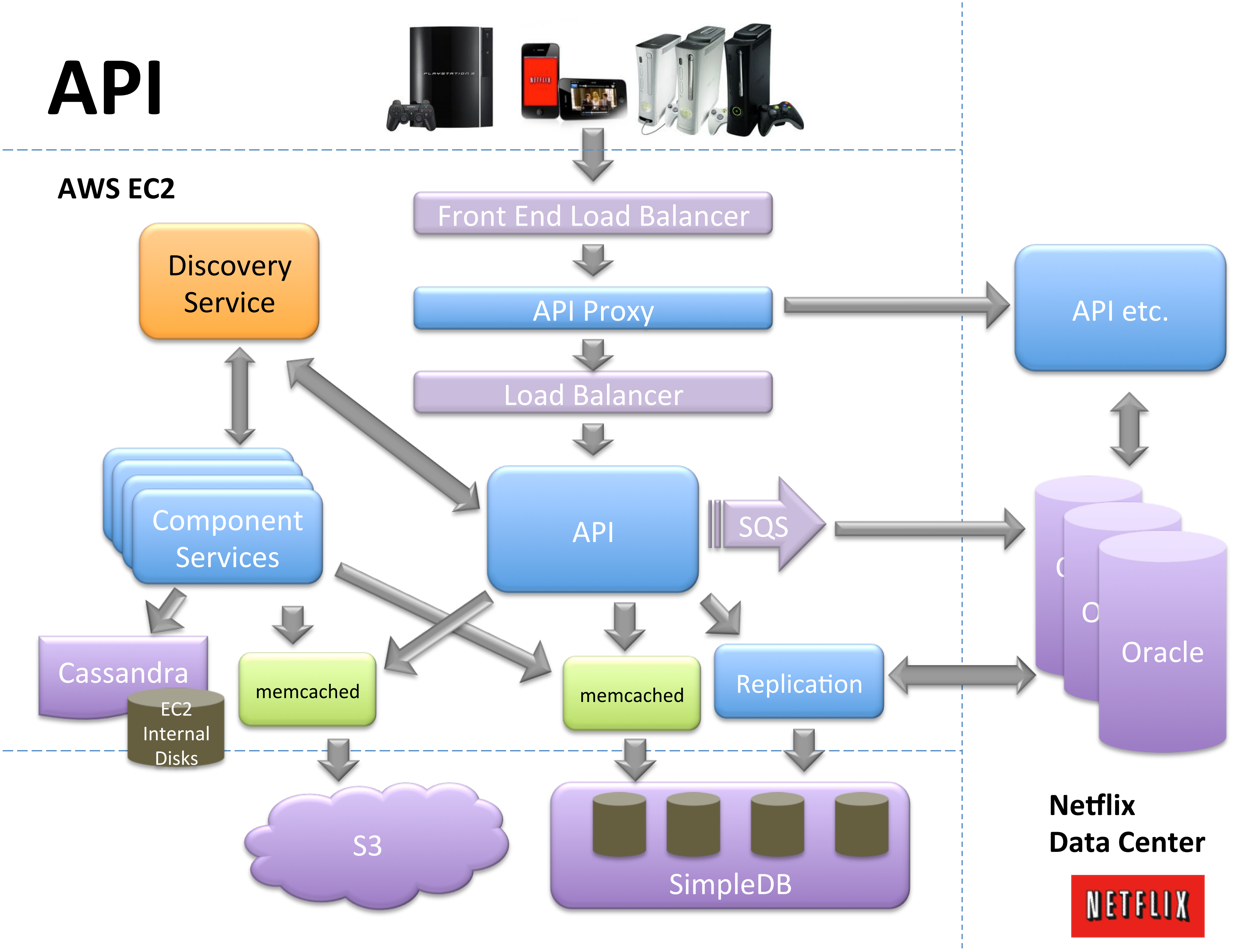
- No schema to change, no scheduled downtime
- Latency for typical queries
  - Memcached is dominated by network latency <1ms
  - Cassandra replication takes a few milliseconds
  - Oracle for simple queries is a few milliseconds
  - SimpleDB replication and REST auth overheads >10ms

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



AWS EC2

Front End Load Balancer

API Proxy

API etc.

Discovery Service

Load Balancer

Component Services

API

SQS

Oracle

Cassandra

EC2 Internal Disks

memcached

memcached

Replication

S3

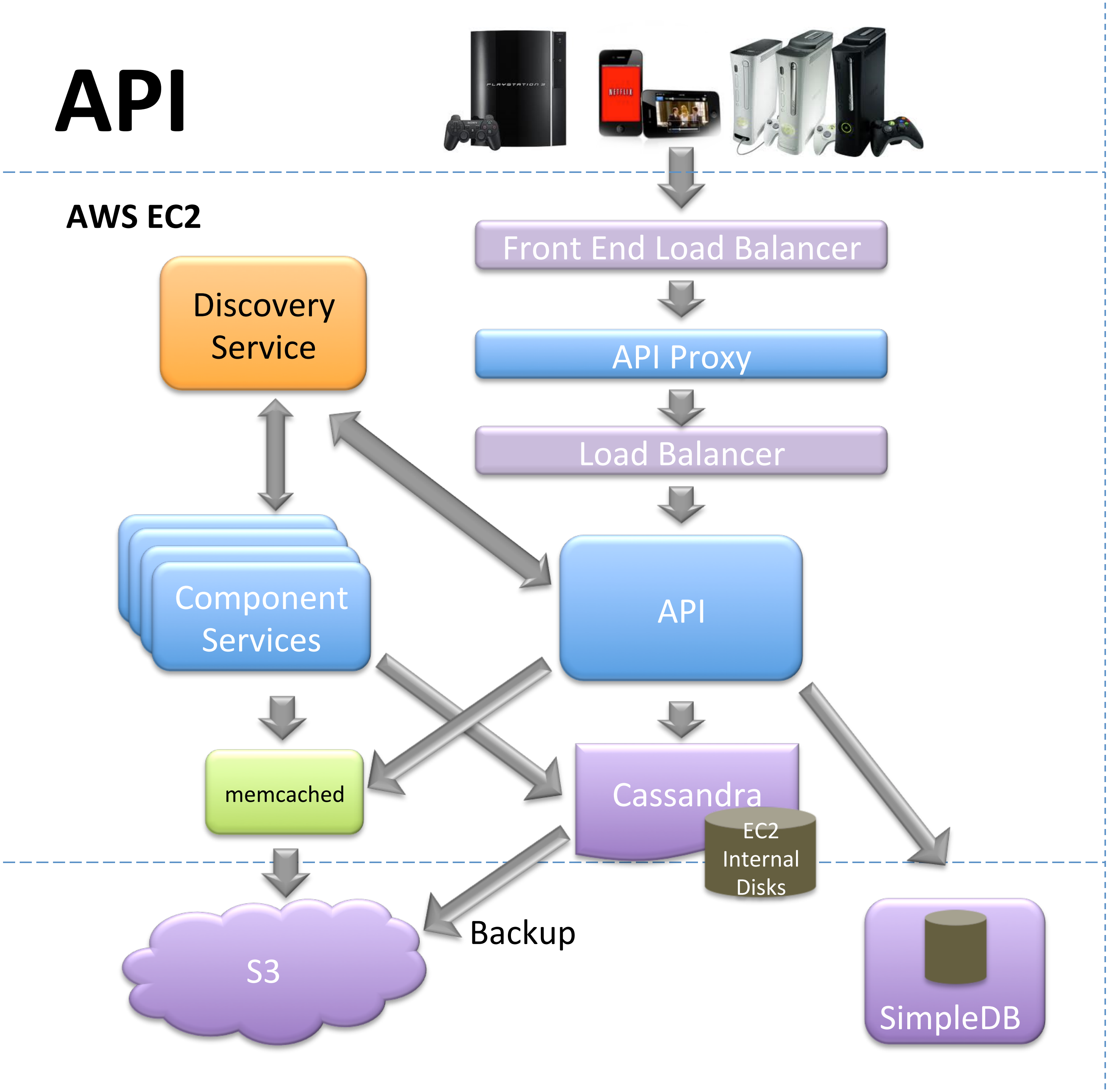
SimpleDB

Netflix Data Center

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 simplest small use cases for now
- New challenges
  - Backup, restore, archive, business continuity
  - Business Intelligence integration

API


AWS EC2
Front End Load Balancer
API Proxy
Load Balancer
Discovery Service
Component Services
API
memcached
Cassandra
EC2 Internal Disks
S3
Backup
SimpleDB

High Availability

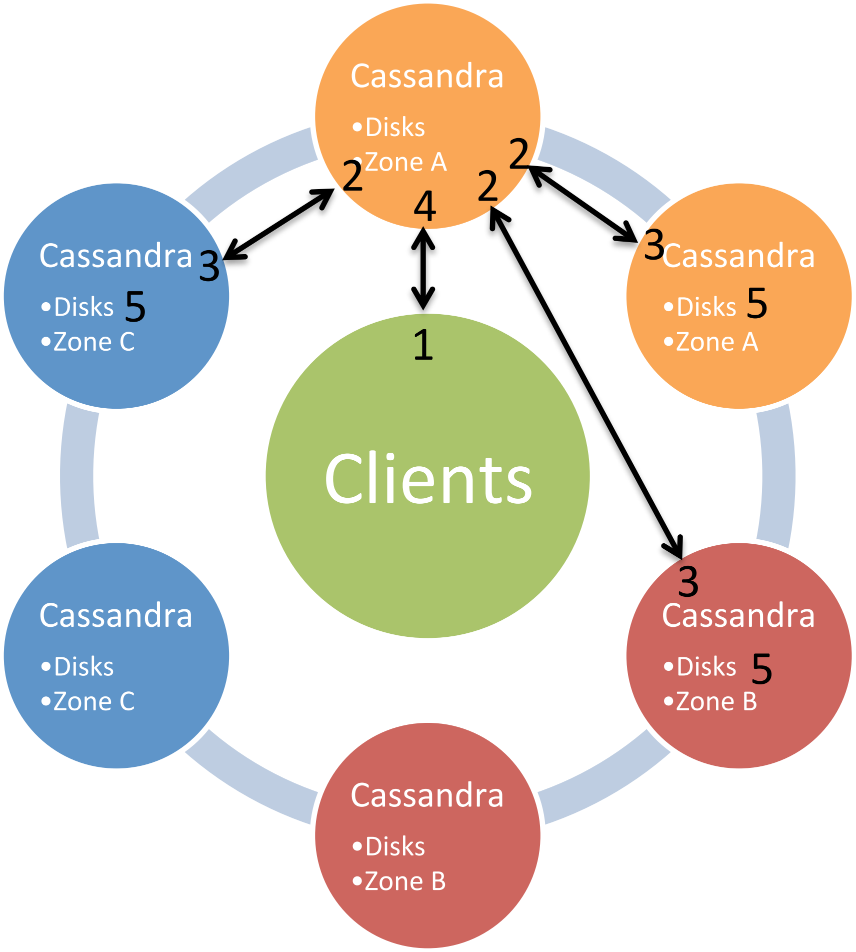
- Cassandra stores 3 local copies, 1 per zone
  - Synchronous access, durable, highly available
  - Read/Write One fastest, least consistent - ~1ms
  - Read/Write Quorum 2 of 3, consistent - ~3ms
- AWS Availability Zones
  - Separate buildings
  - Separate power etc.
  - Close together

NETFLIX

Cassandra Write Data Flows

Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk



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

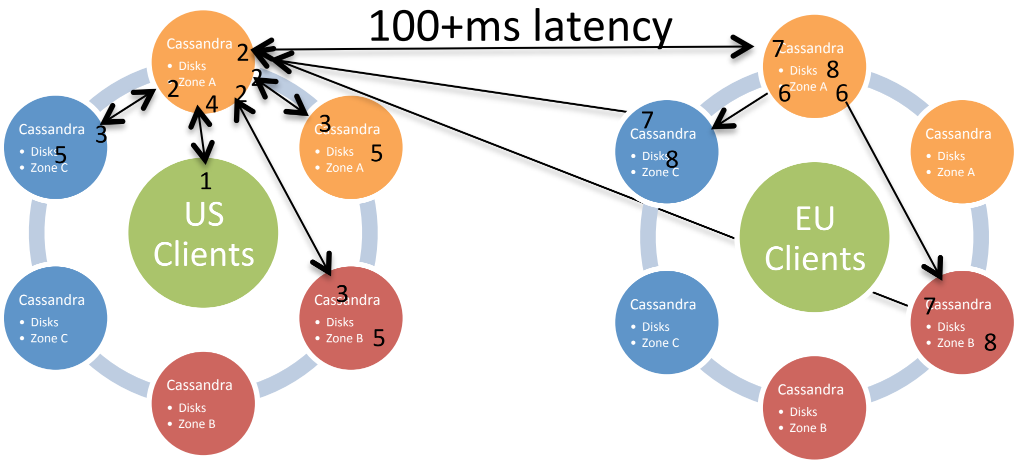
SSTable disk writes and compactions occur asynchronously

Data Flows for Multi-Region Writes

Consistency Level = Local Quorum

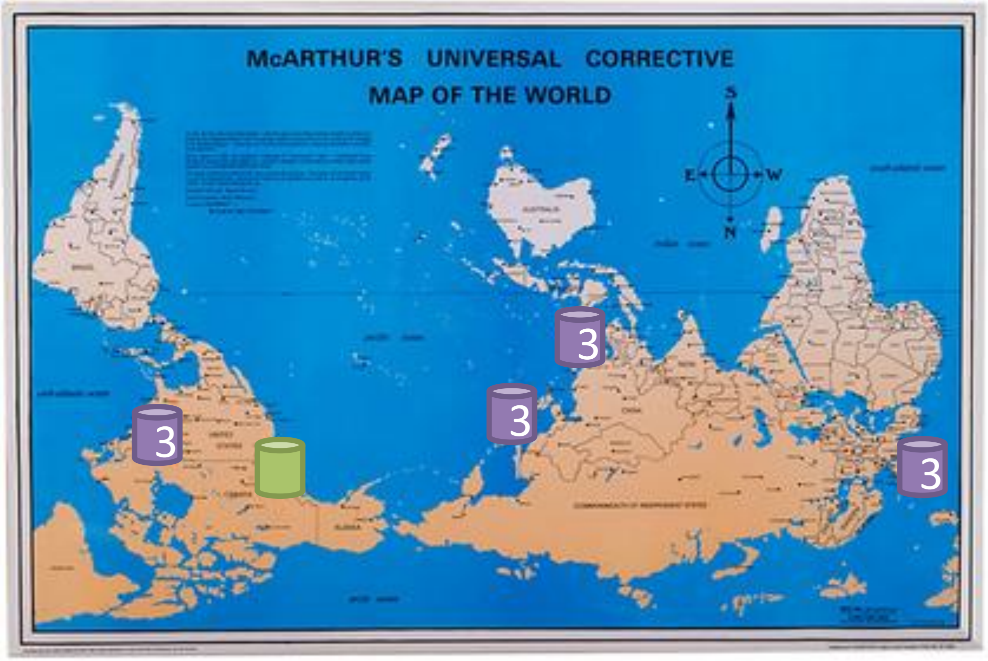
1. Client Writes to any Cassandra Node
2. Coordinator node replicates to other nodes Zones and regions
3. Local write acks returned to coordinator
4. Client gets ack when 2 of 3 local nodes are committed
5. Data written to internal commit log disks
6. When data arrives, remote node replicates data
7. Ack direct to source region coordinator
8. Remote copies written to commit log disks

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.



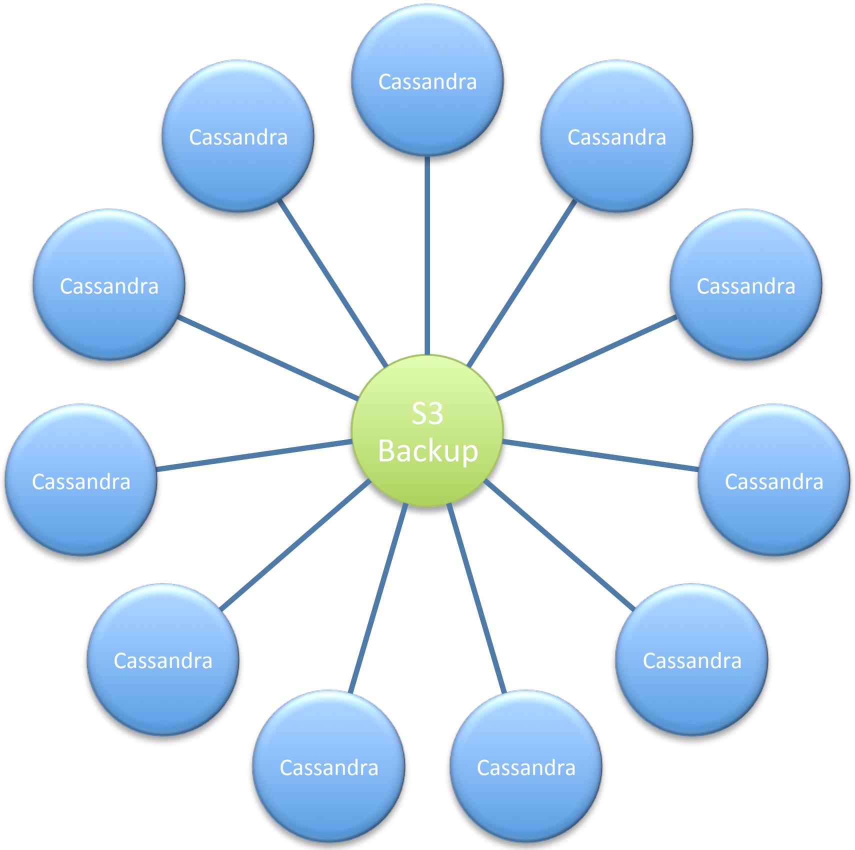
Remote Copies

- 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



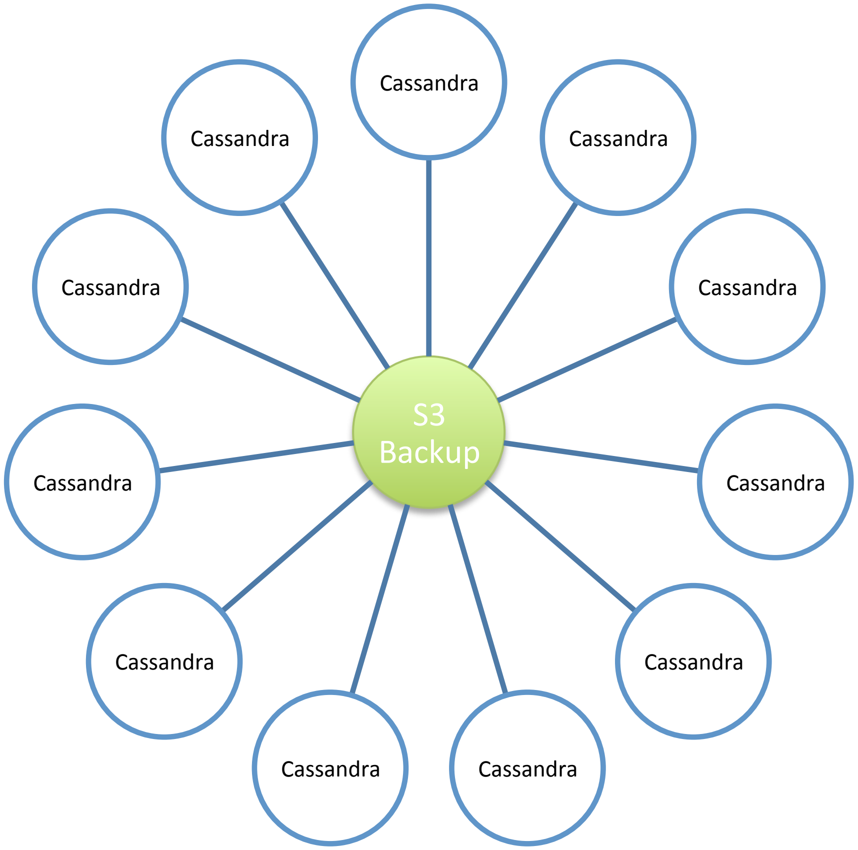
Cassandra Backup

- Full Backup
  - Time based snapshot
  - SSTable compress -> S3
- Incremental
  - SSTable write triggers compressed copy to S3
- Continuous Option
  - Scrape commit log
  - Write to EBS every 30s



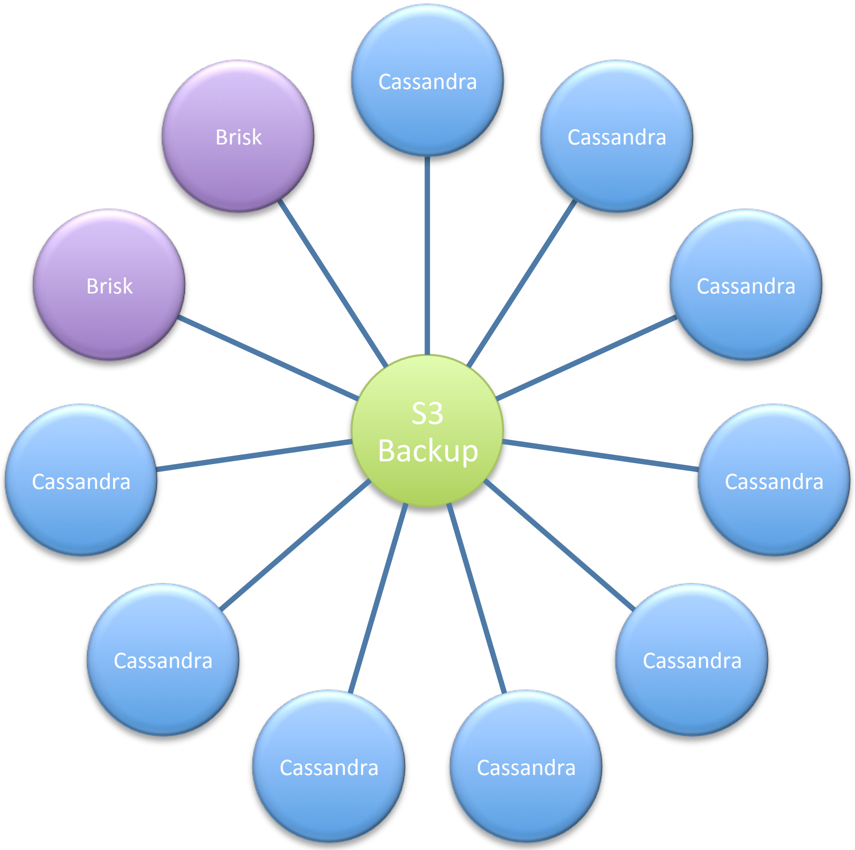
Cassandra Restore

- Full Restore
  - Replace previous data
- New Ring from Backup
  - New name old data
- Scripted
  - Create new instances
  - Parallel load - fast



Cassandra Online Analytics

- Brisk = Hadoop + Cass
  - – Use split Brisk ring
  - – Size each separately
- Direct Access
  - – Keyspaces
  - – Hive/Pig/Map-Reduce
  - – Hdfs as a keyspace
  - – Distributed namenode



Cassandra Archive

Appropriate level of paranoia needed…

- Archive could be un-readable
  - Restore S3 backups weekly from prod to test
- Archive could be stolen
  - PGP 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 5th – 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

Performance and Scalability

Cassandra Use Cases

- Key by Customer – Cross-region clusters
  - Many app specific Cassandra clusters, read-intensive
  - Keys+Rows in memory using m2.4xl Instances
- Key by Customer:Movie – e.g. Viewing History
  - Growing fast, write intensive – m1.xl instances
  - Keys cached in memory, one cluster per region
- Large scale data logging – lots of writes
  - Column data expires after time period
  - Distributed counters, one cluster per region

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

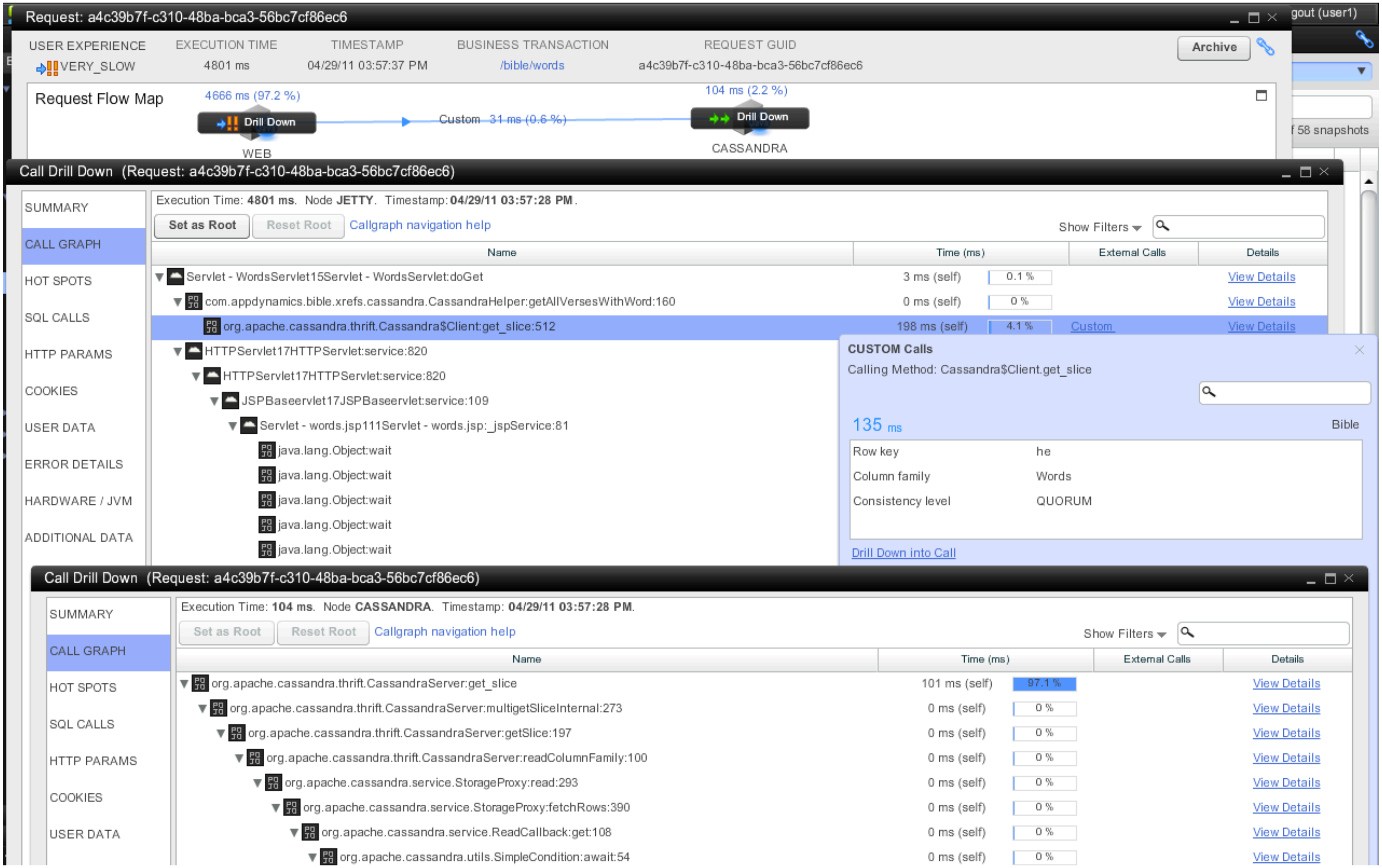
Netflix Platform Cassandra AMI

- Tomcat server
  - Always running, registers with platform
  - Manages Cassandra state, tokens, backups
- Removed Root Disk Dependency on EBS
  - Use S3 backed AMI for stateful services
  - Normally use EBS backed AMI for fast provisioning

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

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

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

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 %		View Details
com.appdynamics.bible.xrefs.cassandra.CassandraHelper:getAllVersesWithWord:160	0 ms (self)	0 %		View Details
org.apache.cassandra.thrift.Cassandra$Client:get_slice:512	198 ms (self)	4.1 %	Custom	View Details
HTTPServlet17HTTPServlet:service:820				
HTTPServlet17HTTPServlet:service:820				
JSPBaseervlet17JSPBaseervlet:service:109				
Servlet - words.jsp111Servlet - words.jsp:_jspService:81				
java.lang.Object:wait				
java.lang.Object:wait				
java.lang.Object:wait				
java.lang.Object:wait				
java.lang.Object:wait				

CUSTOM Calls

Calling Method: Cassandra$Client.get_slice

135 ms Bible

Row key	he
Column family	Words
Consistency level	QUORUM

Drill Down into Call

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

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 %		View Details
org.apache.cassandra.thrift.CassandraServer:multigetSliceInternal:273	0 ms (self)	0 %		View Details
org.apache.cassandra.thrift.CassandraServer:getSlice:197	0 ms (self)	0 %		View Details
org.apache.cassandra.thrift.CassandraServer:readColumnFamily:100	0 ms (self)	0 %		View Details
org.apache.cassandra.service.StorageProxy:read:293	0 ms (self)	0 %		View Details
org.apache.cassandra.service.StorageProxy:fetchRows:390	0 ms (self)	0 %		View Details
org.apache.cassandra.service.ReadCallback:get:108	0 ms (self)	0 %		View Details
org.apache.cassandra.utils.SimpleCondition:await:54	0 ms (self)	0 %		View Details

NETFLIX

Netflix Contributions to Cassandra

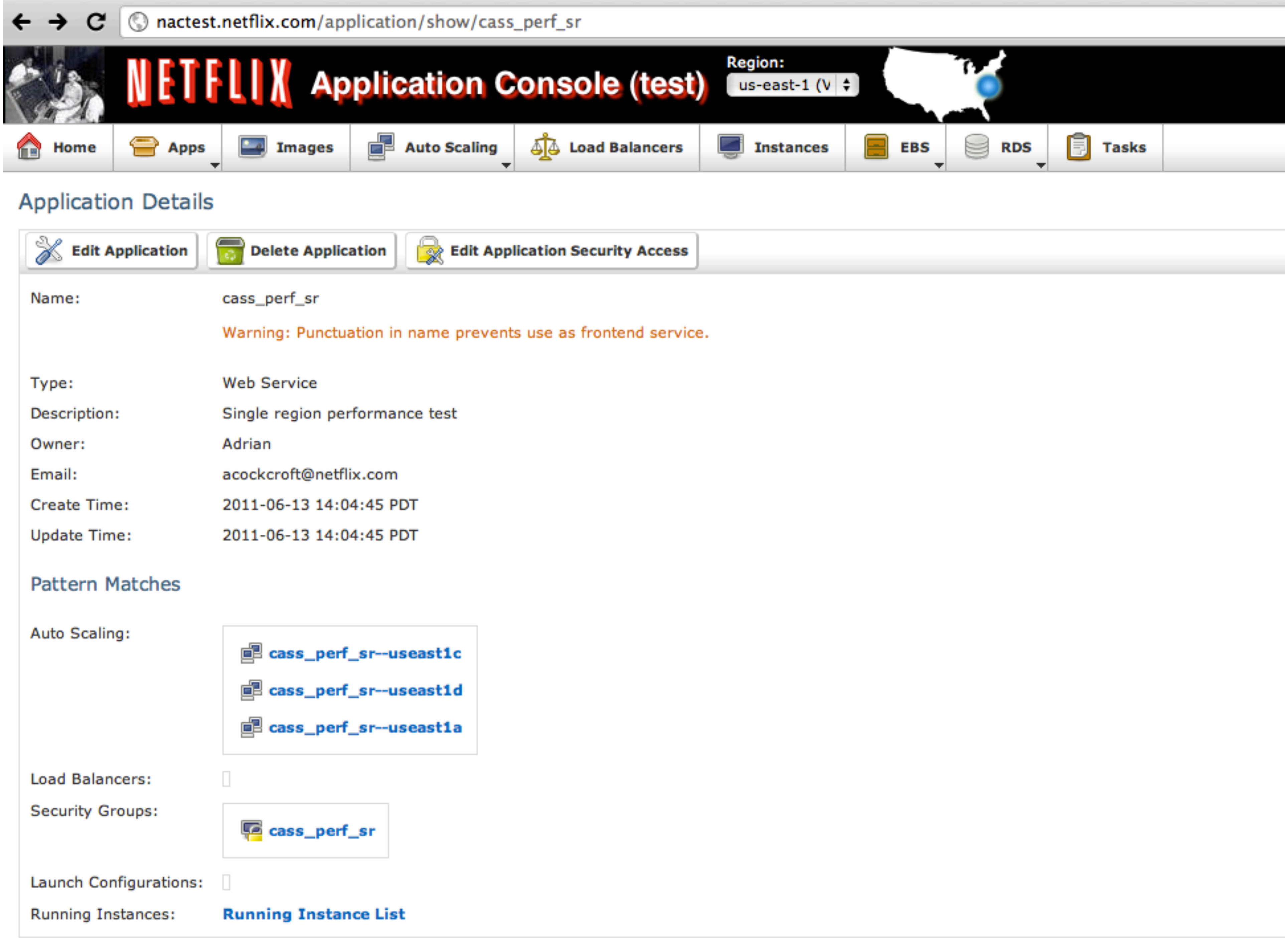
- Cassandra as a mutable toolkit
  - Cassandra is in Java, pluggable, well structured
  - Netflix has a building full of Java engineers….

- Actual Contributions delivered in 0.8
  - First prototype of off-heap row cache
  - Incremental backup SSTable write callback

- Work In Progress
  - AWS integration and backup using Tomcat helper
  - Astyanax re-write of Hector Java client library

Performance Testing

- Cloud Based Testing – frictionless, elastic
  - Create/destroy any sized cluster in minutes
  - Many test scenarios run in parallel
- Test Scenarios
  - Internal app specific tests
  - Simple “stress” tool provided with Cassandra
- Scale test, keep making the cluster bigger
  - Check that tooling and automation works…
  - How many ten column row writes/sec can we do?



Application Details

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

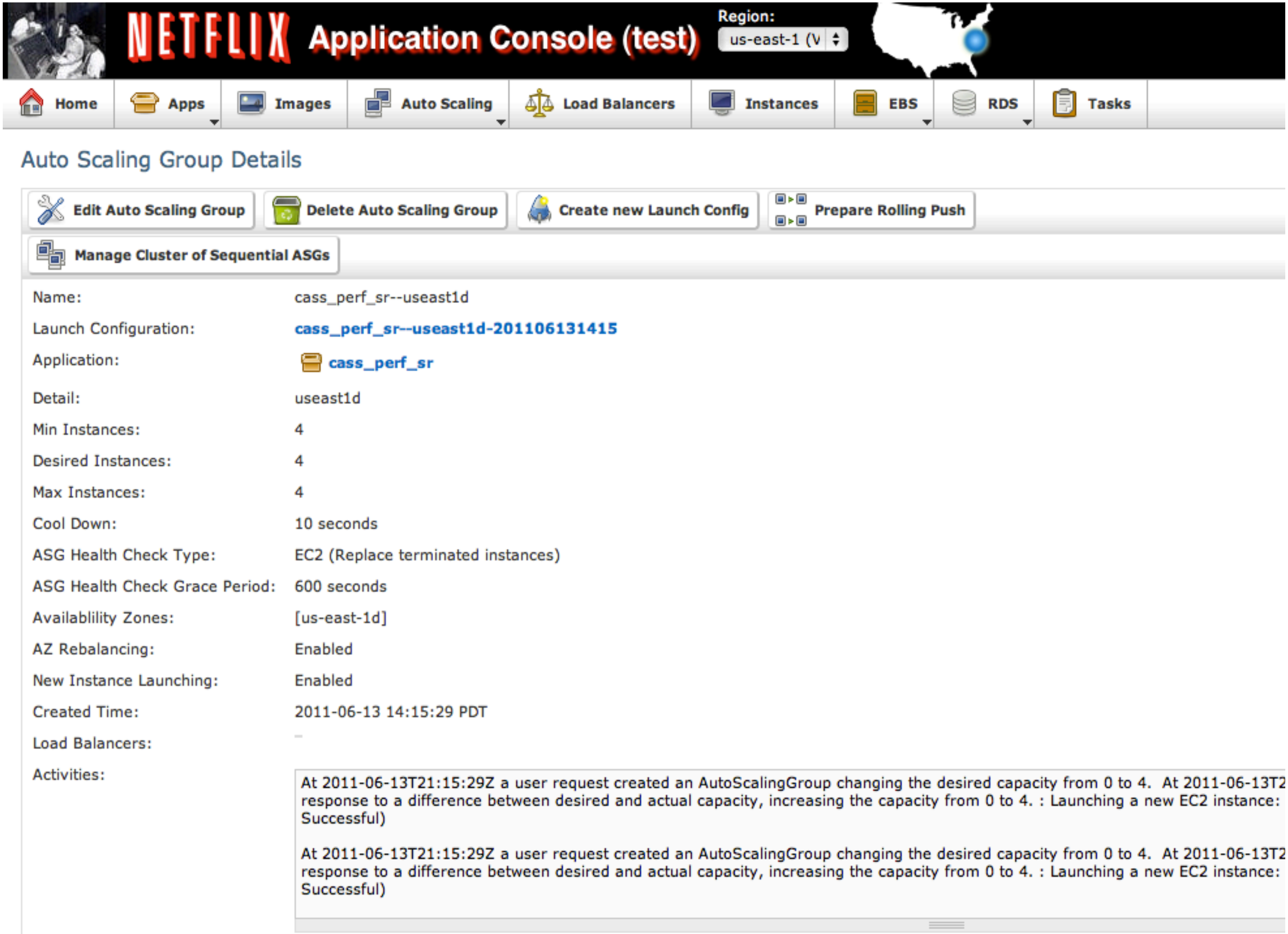
Auto Scaling:
cass_perf_sr--useast1c
cass_perf_sr--useast1d
cass_perf_sr--useast1a

Load Balancers:

Security Groups:
cass_perf_sr

Launch Configurations:

Running Instances: Running Instance List

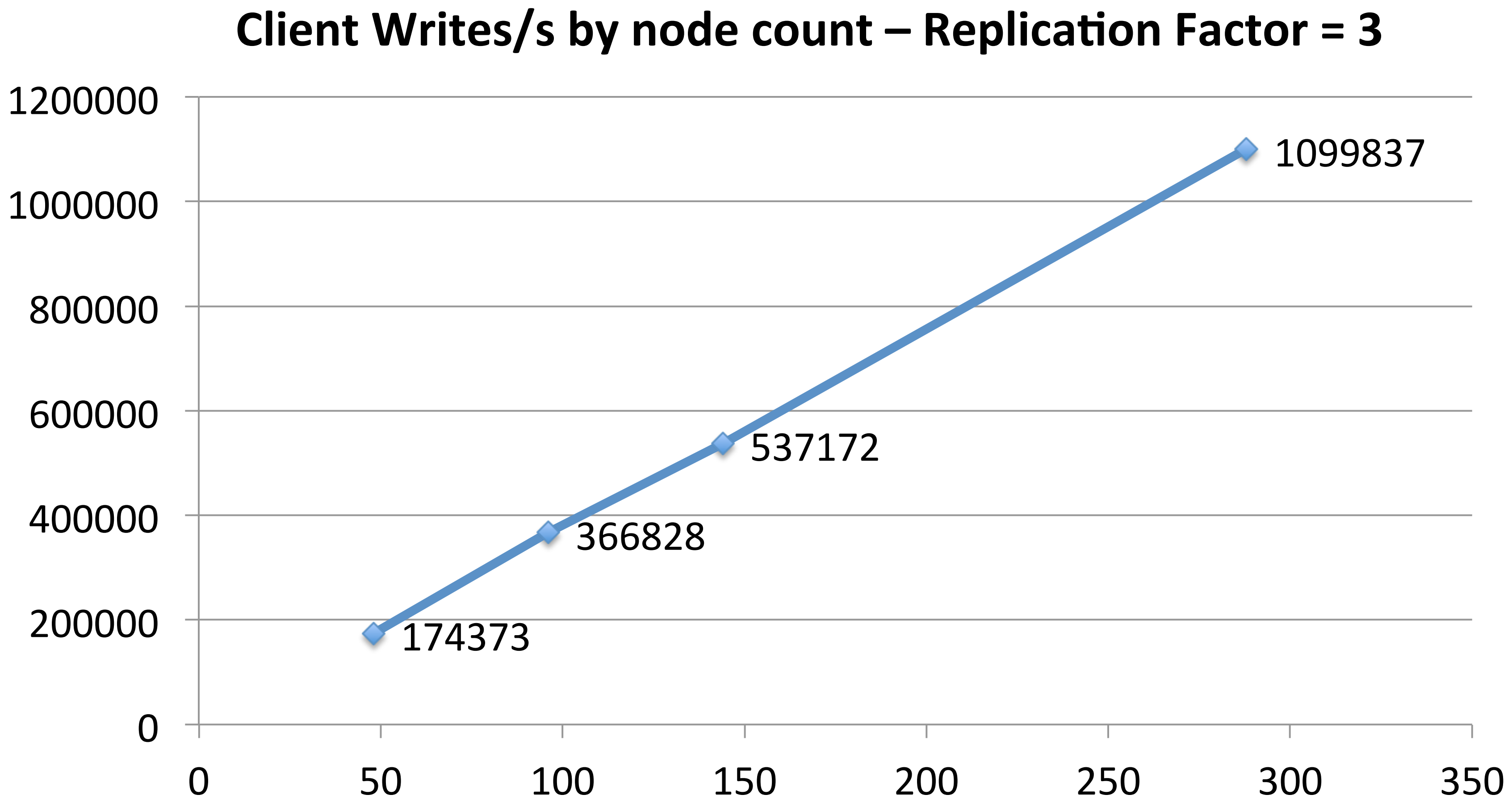


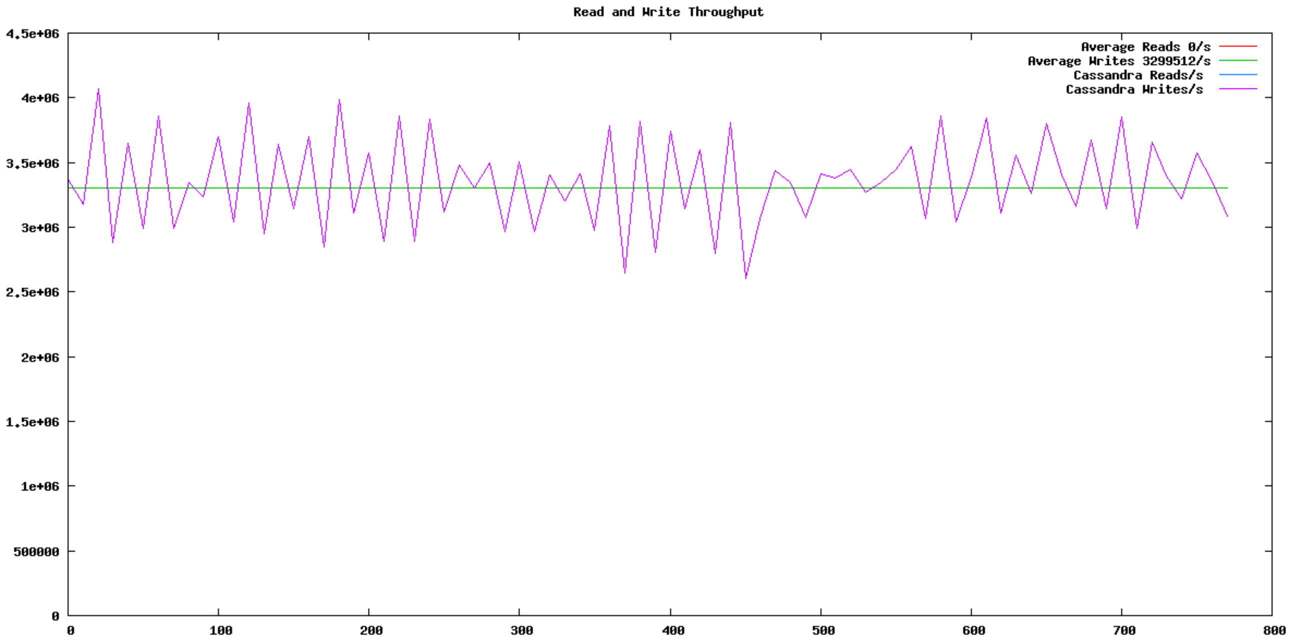
Auto Scaling Group Details

Name:	cass_perf_sr--useast1d
Launch Configuration:	cass_perf_sr--useast1d-201106131415
Application:	cass_perf_sr
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
Availablility Zones:	[us-east-1d]
AZ Rebalancing:	Enabled
New Instance Launching:	Enabled
Created Time:	2011-06-13 14:15:29 PDT
Load Balancers:	
Activities:	At 2011-06-13T21:15:29Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 4. At 2011-06-13T2 response to a difference between desired and actual capacity, increasing the capacity from 0 to 4. : Launching a new EC2 instance: Successful) At 2011-06-13T21:15:29Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 4. At 2011-06-13T2 response to a difference between desired and actual capacity, increasing the capacity from 0 to 4. : Launching a new EC2 instance: Successful)

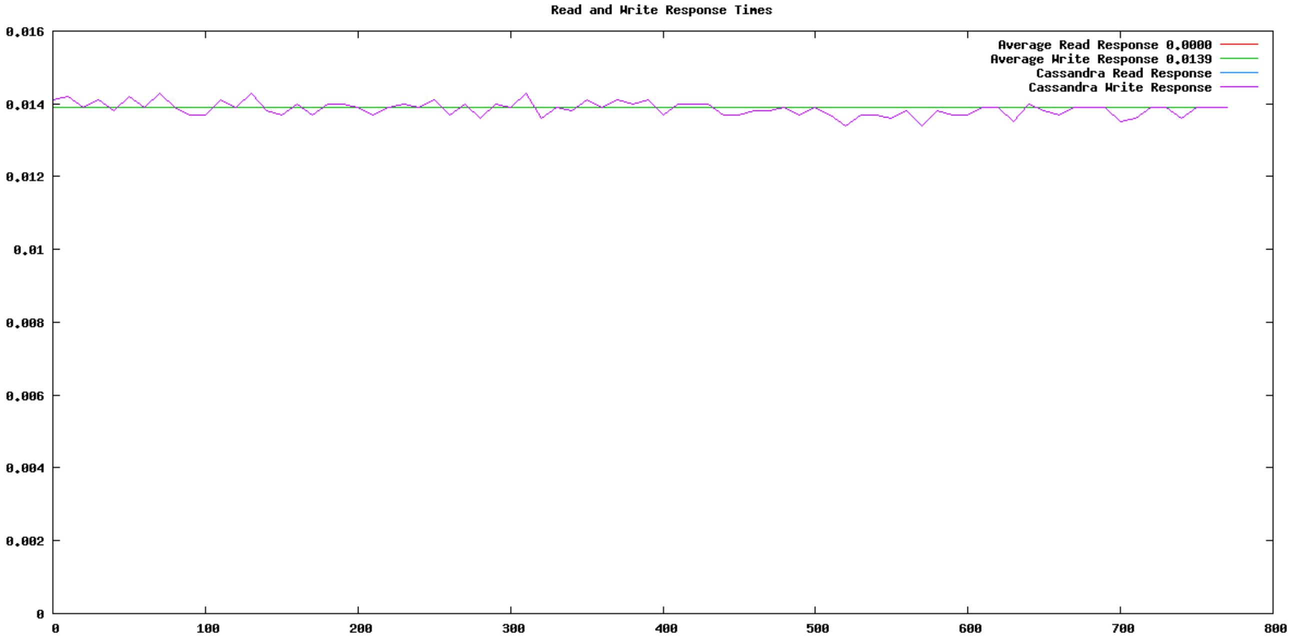
<DrEvil>ONE MILLION</DrEvil>

Scale-Up Linearity

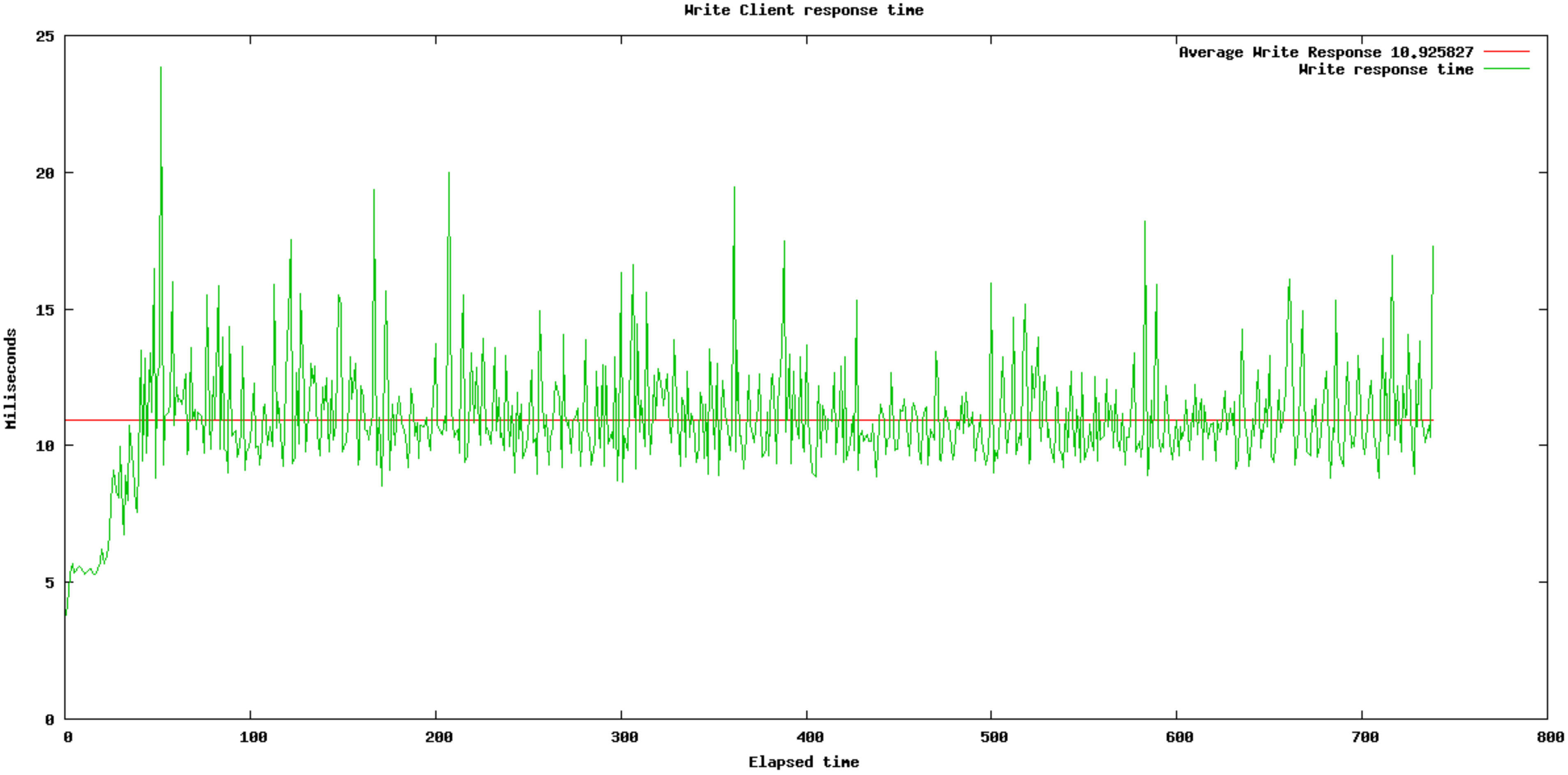




Read and Write Throughput



Read and Write Response Times


Write Client response time
Average Write Response 10.925827
Write response time
Miliseconds
Elapsed time
0
5
10
15
20
25
0
100
200
300
400
500
600
700
800

Per Node Activity

Per Node	48 Nodes	96 Nodes	144 Nodes	288 Nodes
Per Server Writes/s	10,900 w/s	11,460 w/s	11,900 w/s	11,456 w/s
Mean Server Latency	0.0117 ms	0.0134 ms	0.0148 ms	0.0139 ms
Mean CPU %Busy	74.4 %	75.4 %	72.5 %	81.5 %
Disk Read	5,600 KB/s	4,590 KB/s	4,060 KB/s	4,280 KB/s
Disk Write	12,800 KB/s	11,590 KB/s	10,380 KB/s	10,080 KB/s
Network Read	22,460 KB/s	23,610 KB/s	21,390 KB/s	23,640 KB/s
Network Write	18,600 KB/s	19,600 KB/s	17,810 KB/s	19,770 KB/s

Node specification – Xen Virtual Images, AWS US East, three zones

- Cassandra 0.8.6, CentOS, SunJDK6
- AWS EC2 m1 Extra Large – Standard price $ 0.68/Hour
- 15 GB RAM, 4 Cores, 1Gbit network
- 4 internal disks (total 1.6TB, striped together, md, XFS)

NETFLIX

Time is Money

	48 nodes	96 nodes	144 nodes	288 nodes
Writes Capacity	174373 w/s	366828 w/s	537172 w/s	1,099,837 w/s
Storage Capacity	12.8 TB	25.6 TB	38.4 TB	76.8 TB
Nodes Cost/hr	$32.64	$65.28	$97.92	$195.84
Test Driver Instances	10	20	30	60
Test Driver Cost/hr	$20.00	$40.00	$60.00	$120.00
Cross AZ Traffic	5 TB/hr	10 TB/hr	15 TB/hr	30[1] TB/hr
Traffic Cost/10min	$8.33	$16.66	$25.00	$50.00
Setup Duration	15 minutes	22 minutes	31 minutes	66[2] minutes
AWS Billed Duration	1hr	1hr	1 hr	2 hr
Total Test Cost	$60.97	$121.94	$182.92	$561.68

[1] Estimate two thirds of total network traffic

[2] Workaround for a tooling bug slowed setup

Takeaway

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

Also, benchmarking in the cloud is fast, cheap and scalable

http://www.linkedin.com/in/adriancockcroft
@adrianco #netflixcloud
acockcroft@netflix.com

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)