Architectural Patterns for High ~~Anxiety~~ Availability

November 2012

Adrian Cockcroft

@adrianco #netflixcloud #qconsf

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

NETFLIX

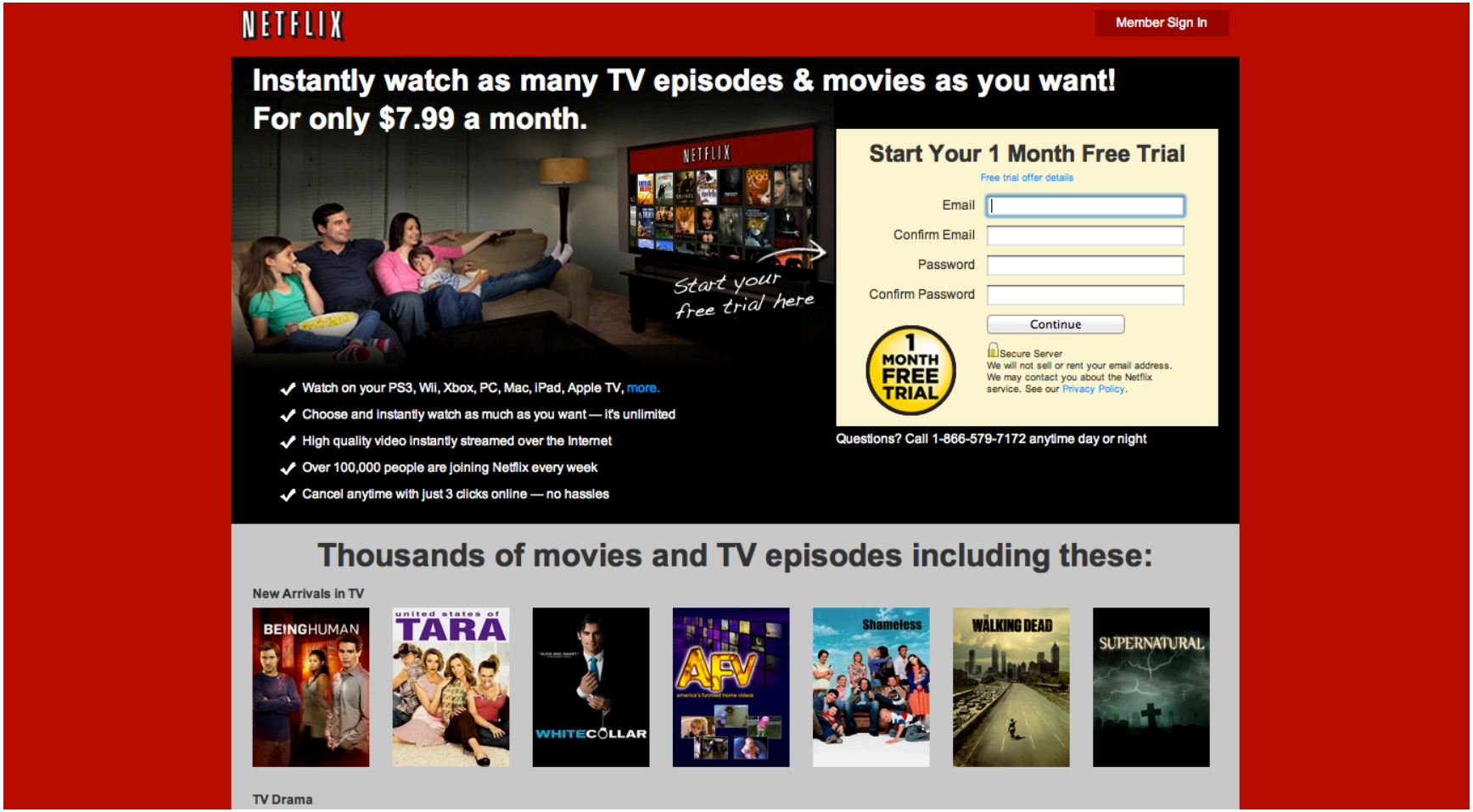
The Netflix Streaming Service

Now in USA, Canada, Latin America,
UK, Ireland, Sweden, Denmark,
Norway and Finland

@adrianco

US Non-Member Web Site

Advertising and Marketing Driven

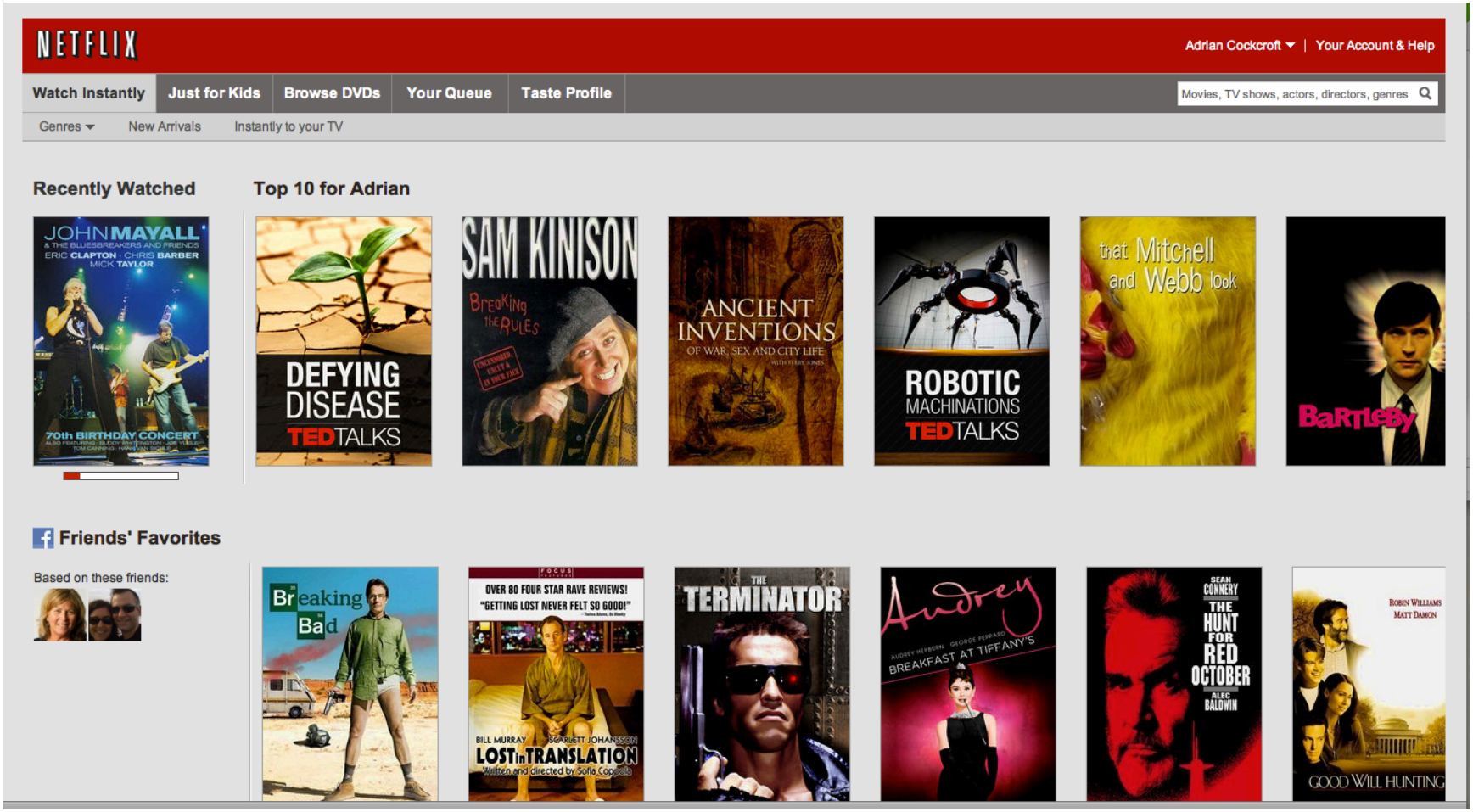


@adrianco

NETFLIX

Member Web Site

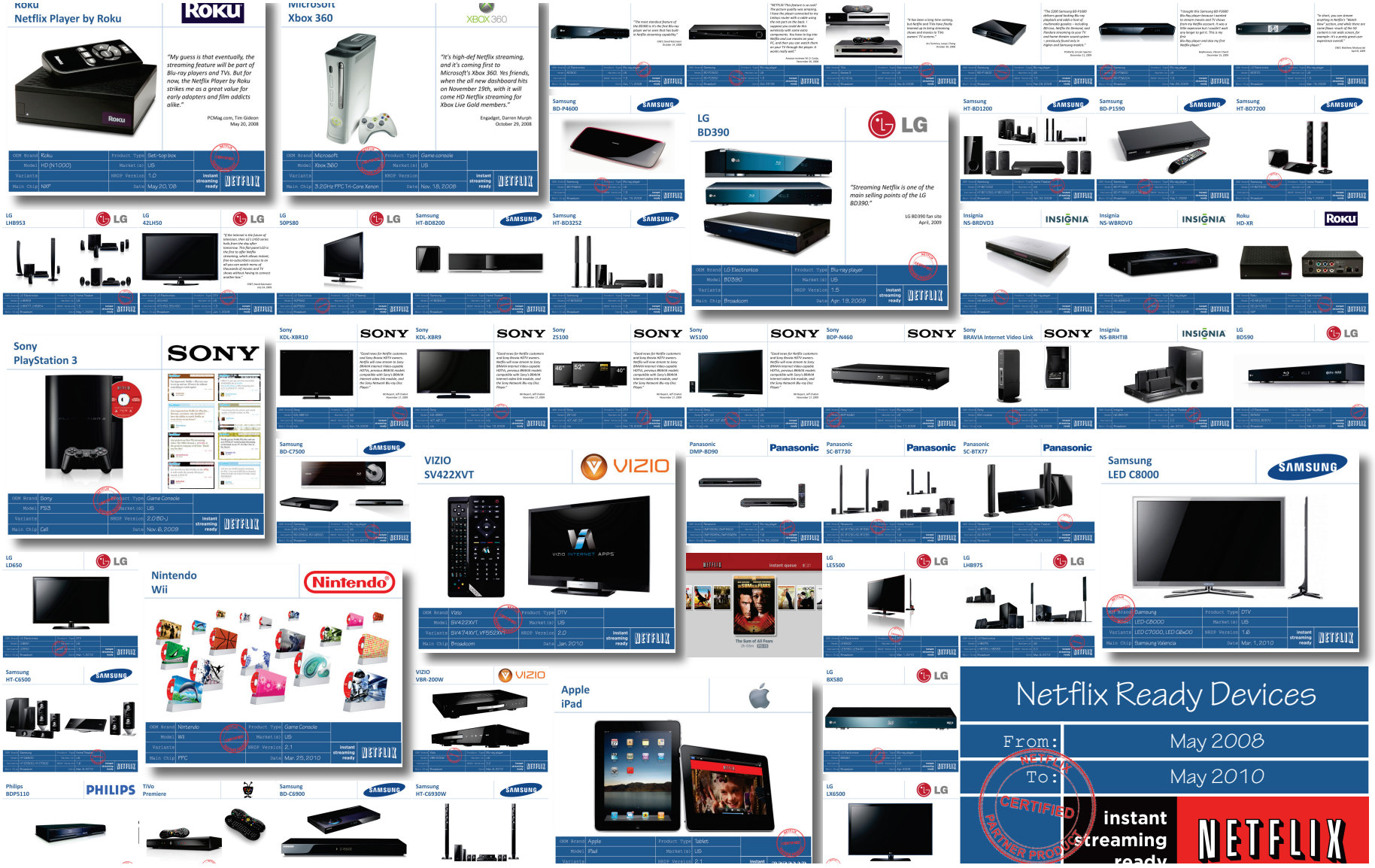
Personalization Driven



@adrianco

NETFLIX

Streaming Device API



Content Delivery Service

Distributed storage nodes controlled by Netflix cloud services

NETFLIX

Open Connect

- Overview
- FAQ
- Peering Information
- > Hardware Design
- Software Design
- Deployment Guide
- ISP Inquiry

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.

The following system was developed and first deployed at the end of 2011

@adrianco

NETFLIX

November 2012 Traffic

	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)

@adrianco

NETFLIX

Abstract

- ~~Netflix on Cloud – What, Why and When~~
- Globally Distributed Architecture
- Benchmarks and Scalability
- Open Source Components
- High Anxiety

NETFLIX

Blah Blah  Blah

(I'm skipping all the cloud intro etc. did that last year… Netflix runs in the cloud, if you hadn't figured that out already you aren't paying attention and should go read Infoq and slideshare.net/netflix)

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

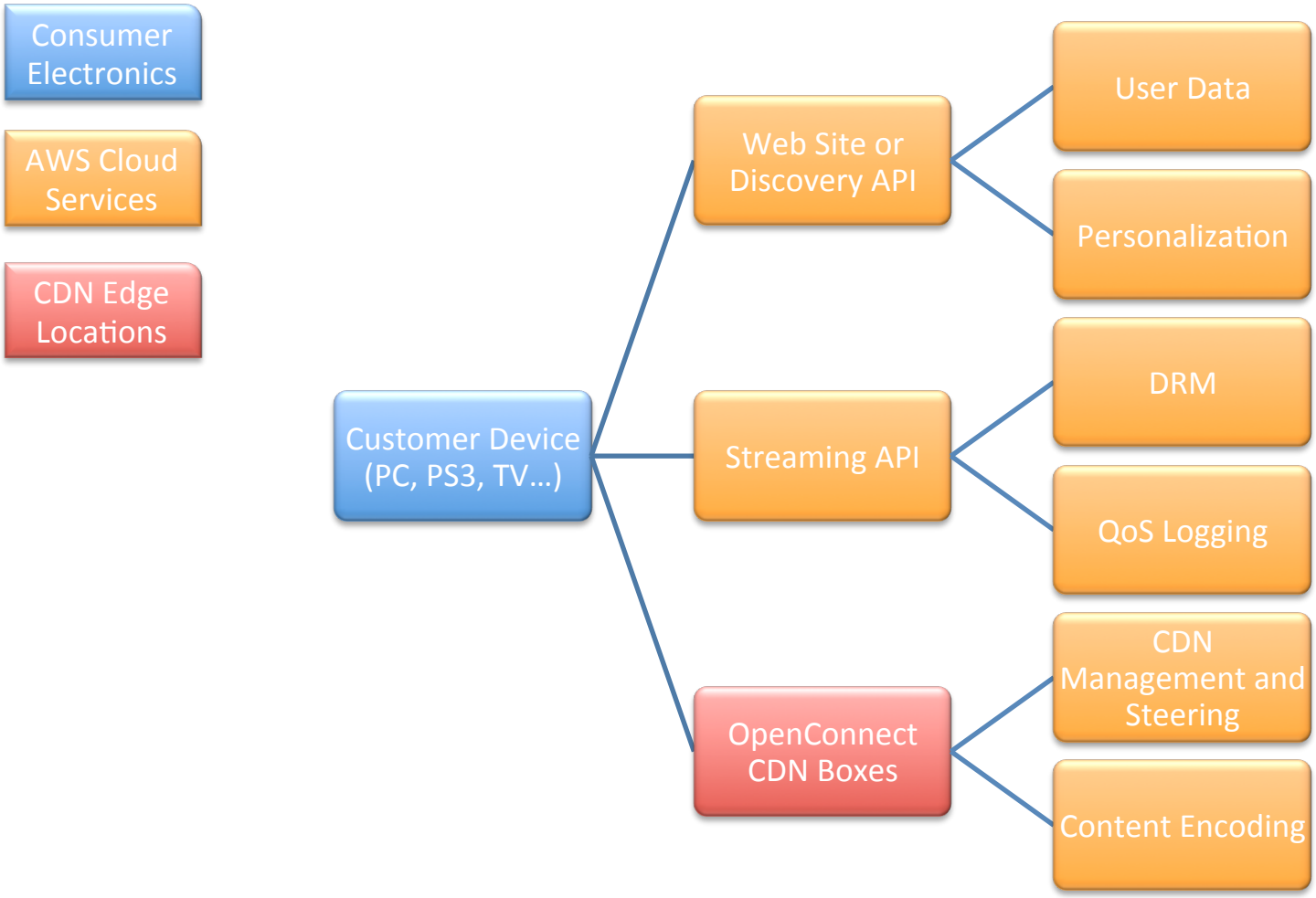
NETFLIX

Things We Do Do…

	In production at Netflix
• Big Data/Hadoop	2009
• AWS Cloud	2009
• Application Performance Management	2010
• Integrated DevOps Practices	2010
• Continuous Integration/Delivery	2010
• NoSQL, Globally Distributed	2010
• Platform as a Service; Micro-Services	2010
• Social coding, open development/github	2011

NETFLIX

How Netflix Works



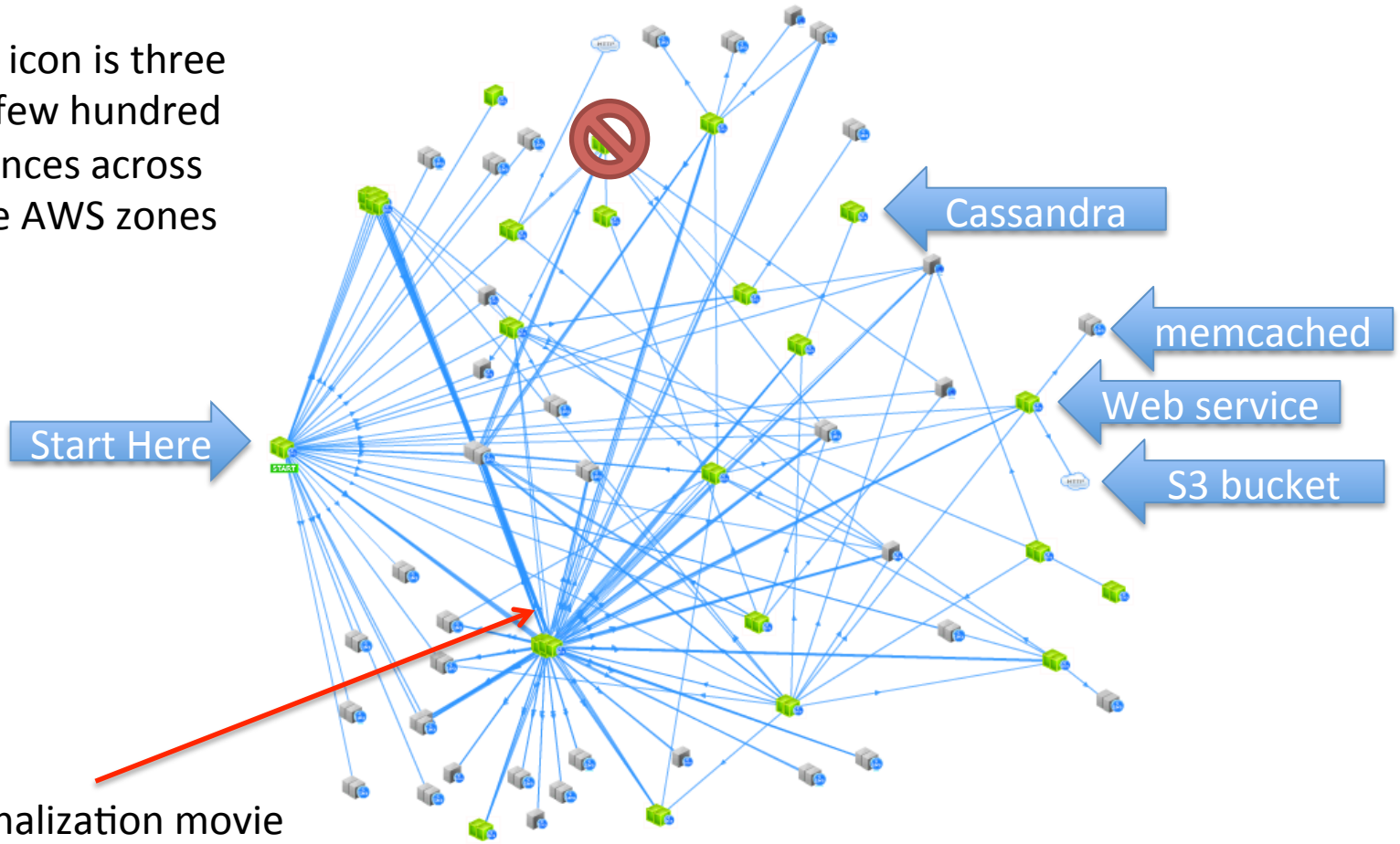
@adrianco

NETFLIX

Web Server Dependencies Flow

(Home page business transaction as seen by AppDynamics)

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



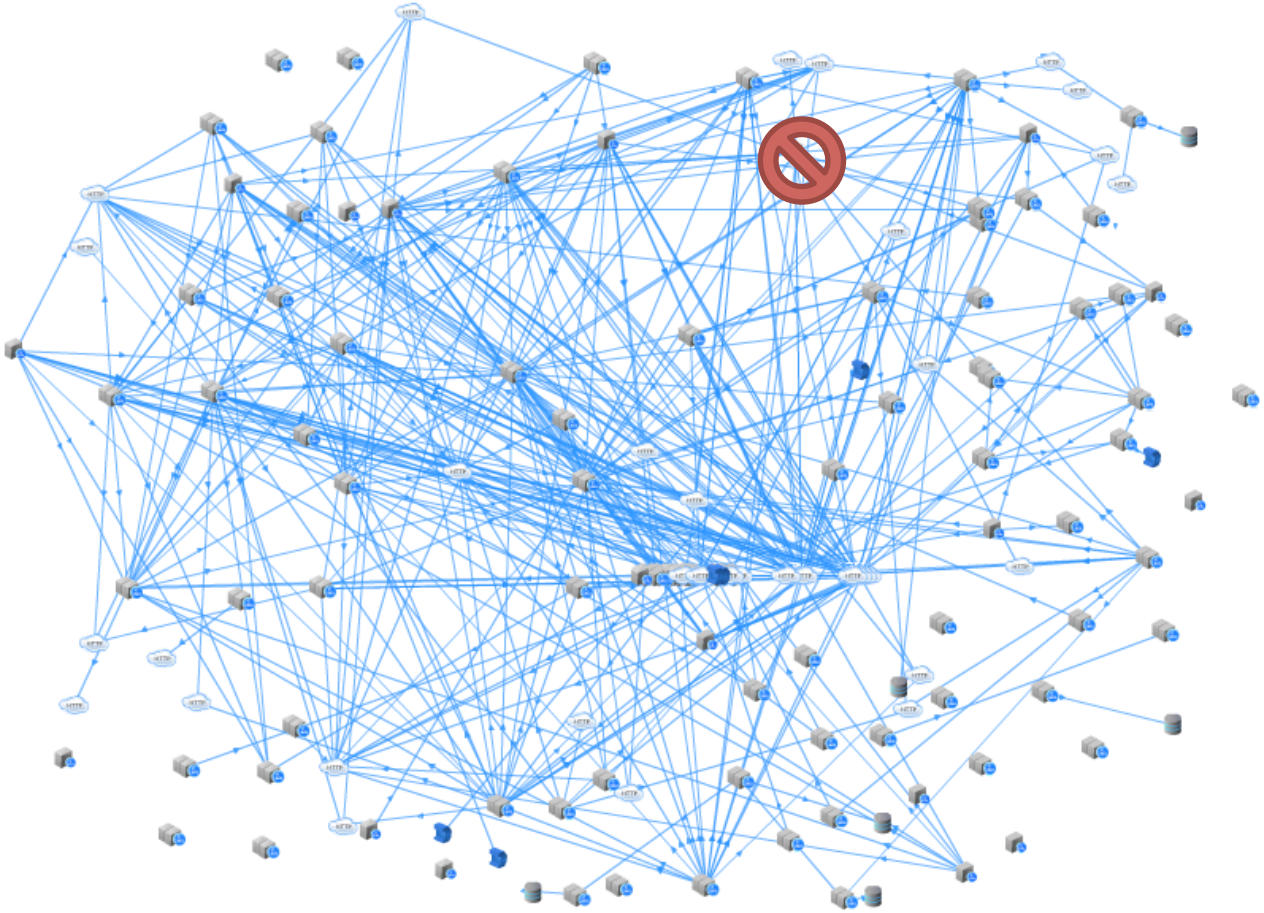
Personalization movie group chooser

@adrianco

NETFLIX

Component Micro-Services

Test With Chaos Monkey, Latency Monkey

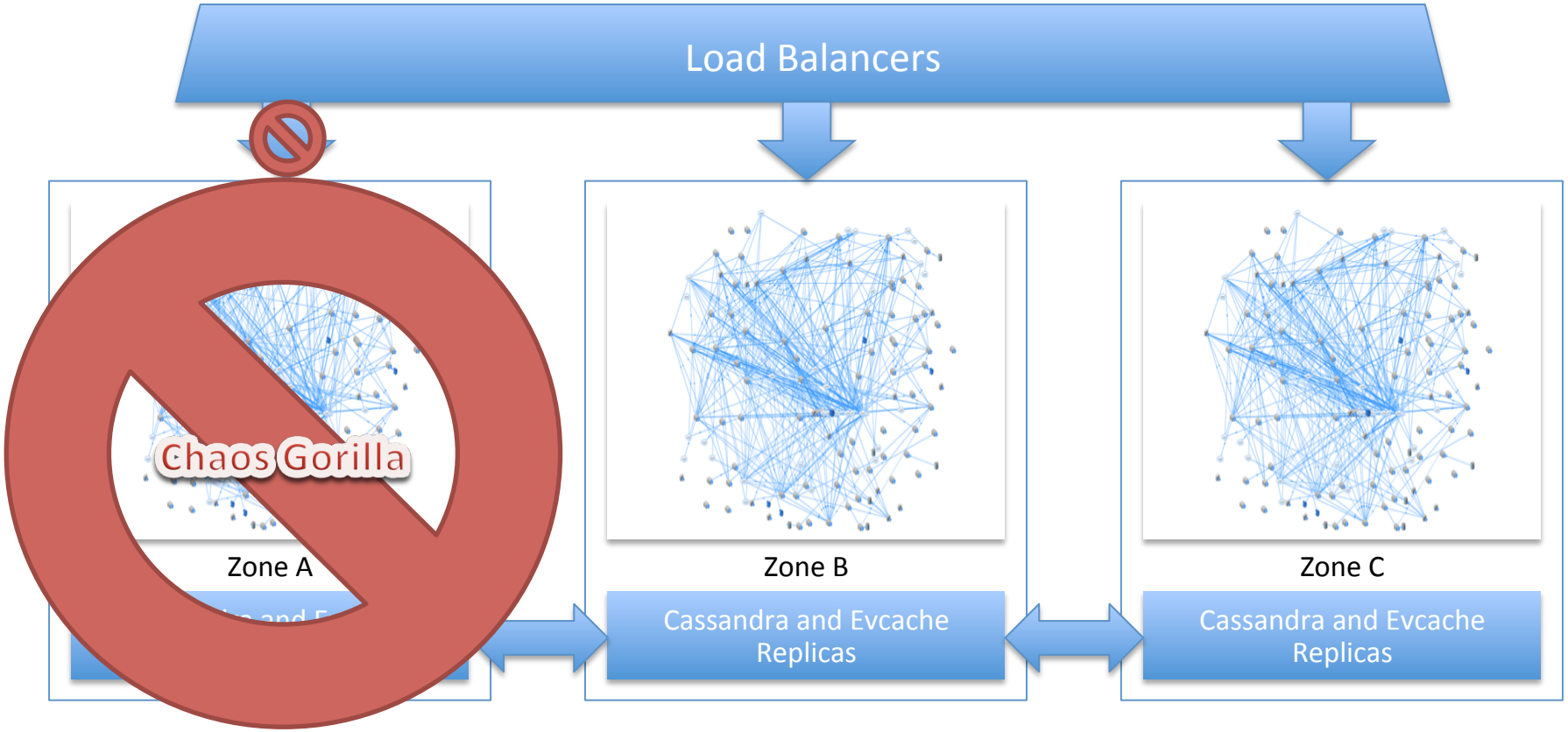


@adrianco

NETFLIX

Three Balanced Availability Zones

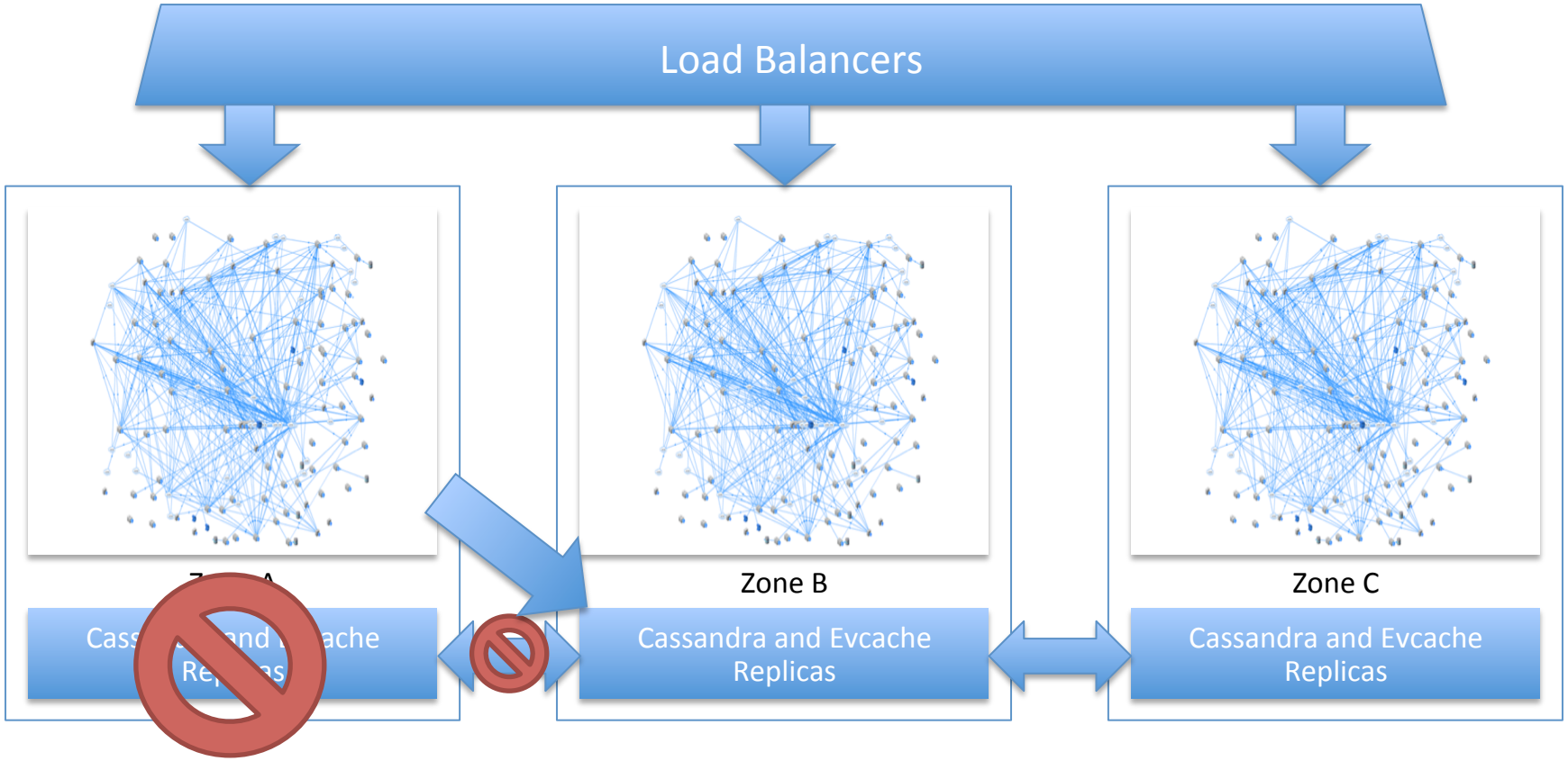
Test with Chaos Gorilla



NETFLIX

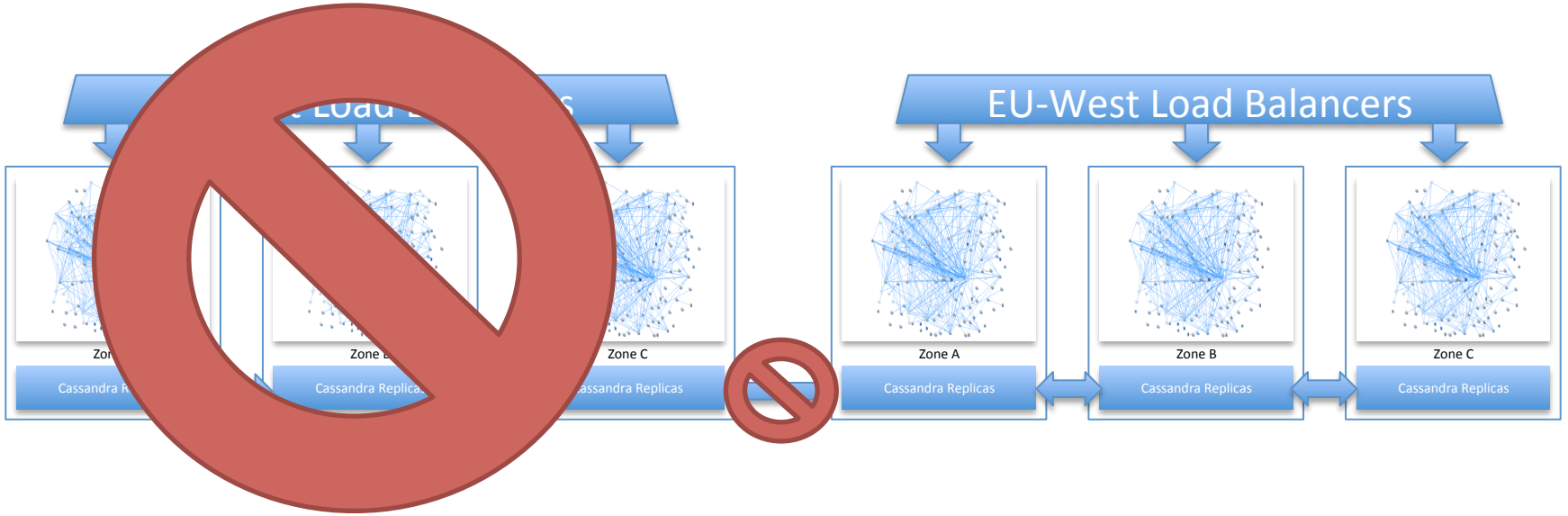
Triple Replicated Persistence

Cassandra maintenance affects individual replicas



NETFLIX

Isolated Regions



NETFLIX

Failure Modes and Effects

Failure Mode	Probability	Mitigation Plan
Application Failure	High	Automatic degraded response
AWS Region Failure	Low	Wait for region to recover
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

Zone Failure Modes

- Power Outage
  - Instances lost, ephemeral state lost
  - Clean break and recovery, fail fast, “no route to host”
- Network Outage
  - Instances isolated, state inconsistent
  - More complex symptoms, recovery issues, transients
- Dependent Service Outage
  - Cascading failures, misbehaving instances, human errors
  - Confusing symptoms, recovery issues, byzantine effects

More detail on this topic at AWS Re:Invent later this month…

NETFLIX

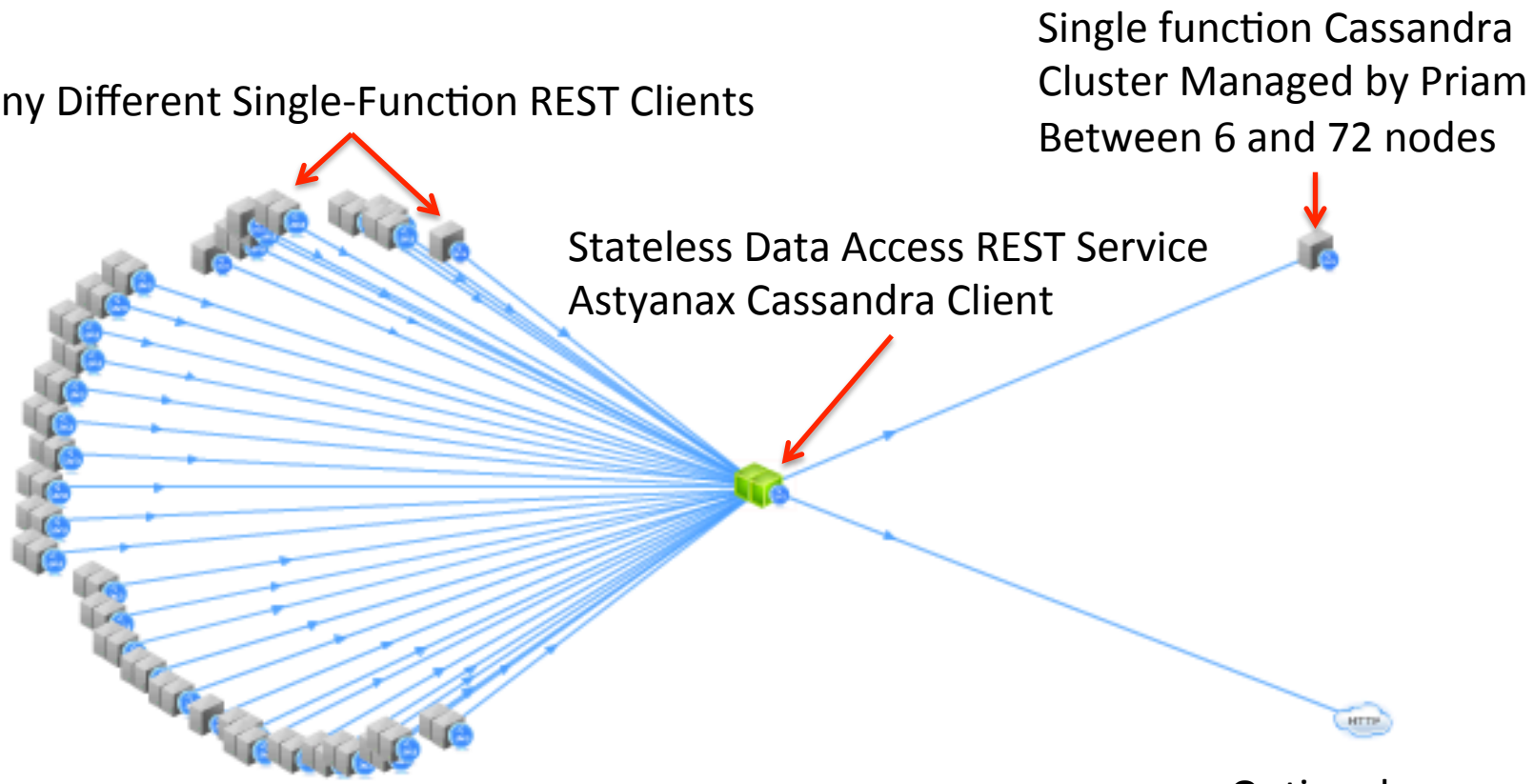
Cassandra backed Micro-Services

A highly scalable, available and durable deployment pattern

@adrianco

Micro-Service Pattern

One keyspace, replaces a single table or materialized view

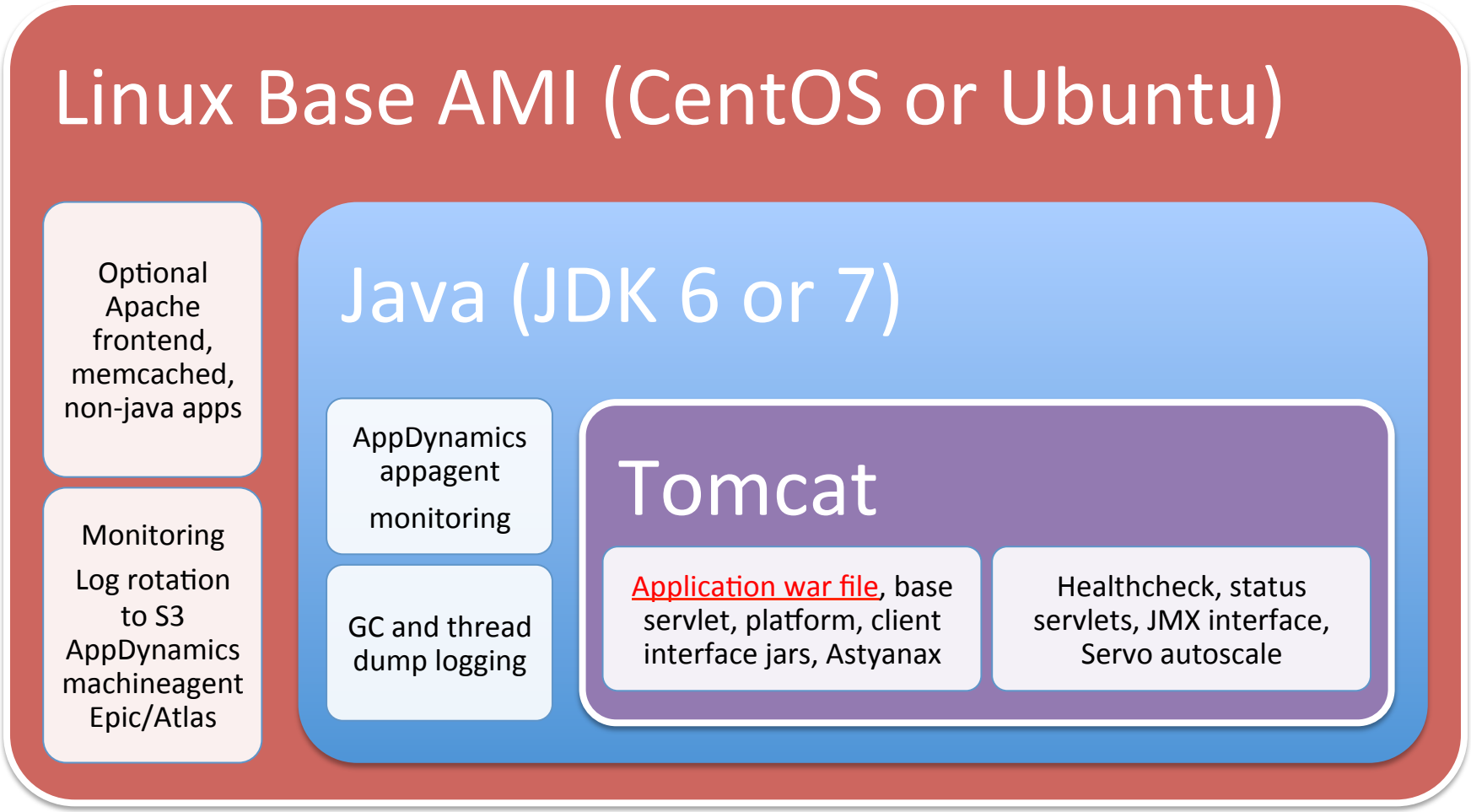


Each icon represents a horizontally scaled service of three to hundreds of instances deployed over three availability zones

Appdynamics Service Flow Visualization

NETFLIX

Stateless Micro-Service Architecture



NETFLIX

Astyanax

Available at http://github.com/netflix

- Features
  - Complete abstraction of connection pool from RPC protocol
  - Fluent Style API
  - Operation retry with backoff
  - Token aware
- Recipes
  - Distributed row lock (without zookeeper)
  - Multi-DC row lock
  - Uniqueness constraint
  - Multi-row uniqueness constraint
  - Chunked and multi-threaded large file storage

Astyanax Query Example

Paginate through all columns in a row

```
ColumnList<String> columns;
int pageize = 10;
try {
    RowQuery<String, String> query = keyspace
        .prepareQuery(CF_STANDARD1)
        .getKey("A")
        .setIsPaginating()
        .withColumnRange(new RangeBuilder().setMaxSize(pageize).build());

    while (!(columns = query.execute().getResult()).isEmpty()) {
        for (Column<String> c : columns) {
        }
    }
} catch (ConnectionException e) {
}
```

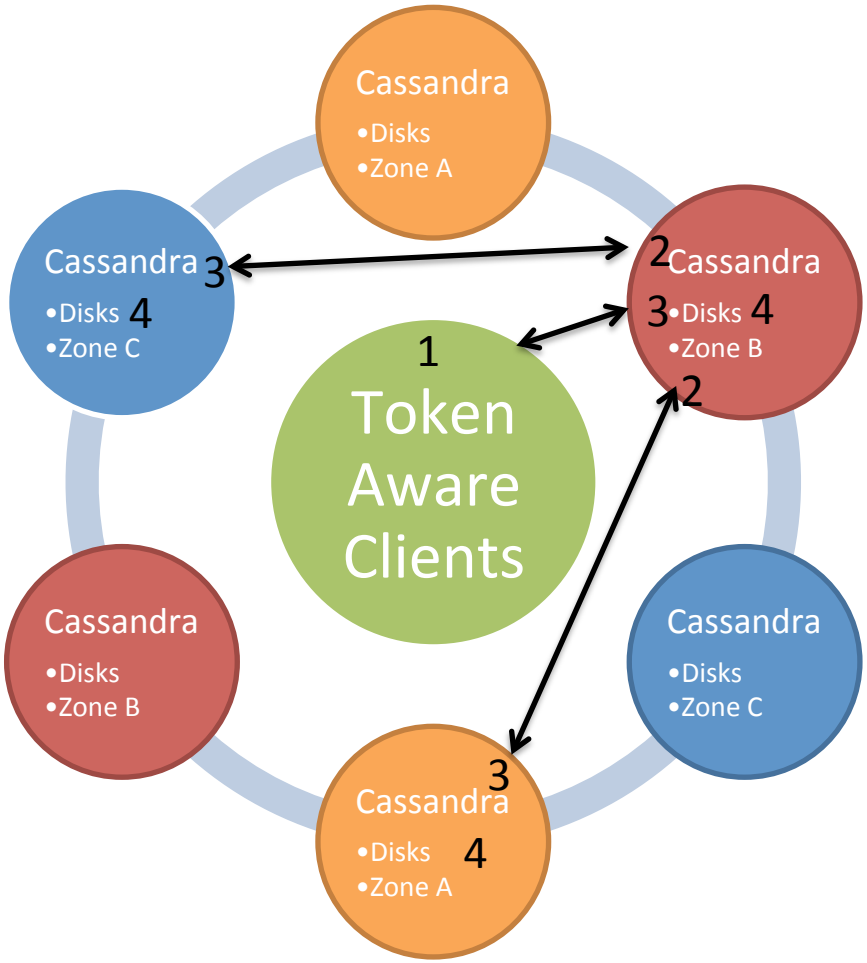
@adrianco

NETFLIX

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

@adrianco

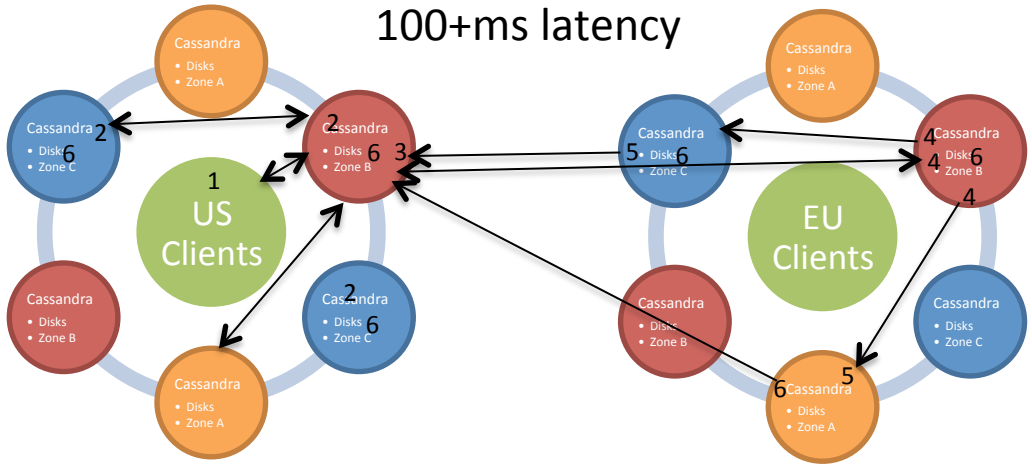
NETFLIX

Data Flows for Multi-Region Writes

Token Aware, Consistency Level = Local Quorum

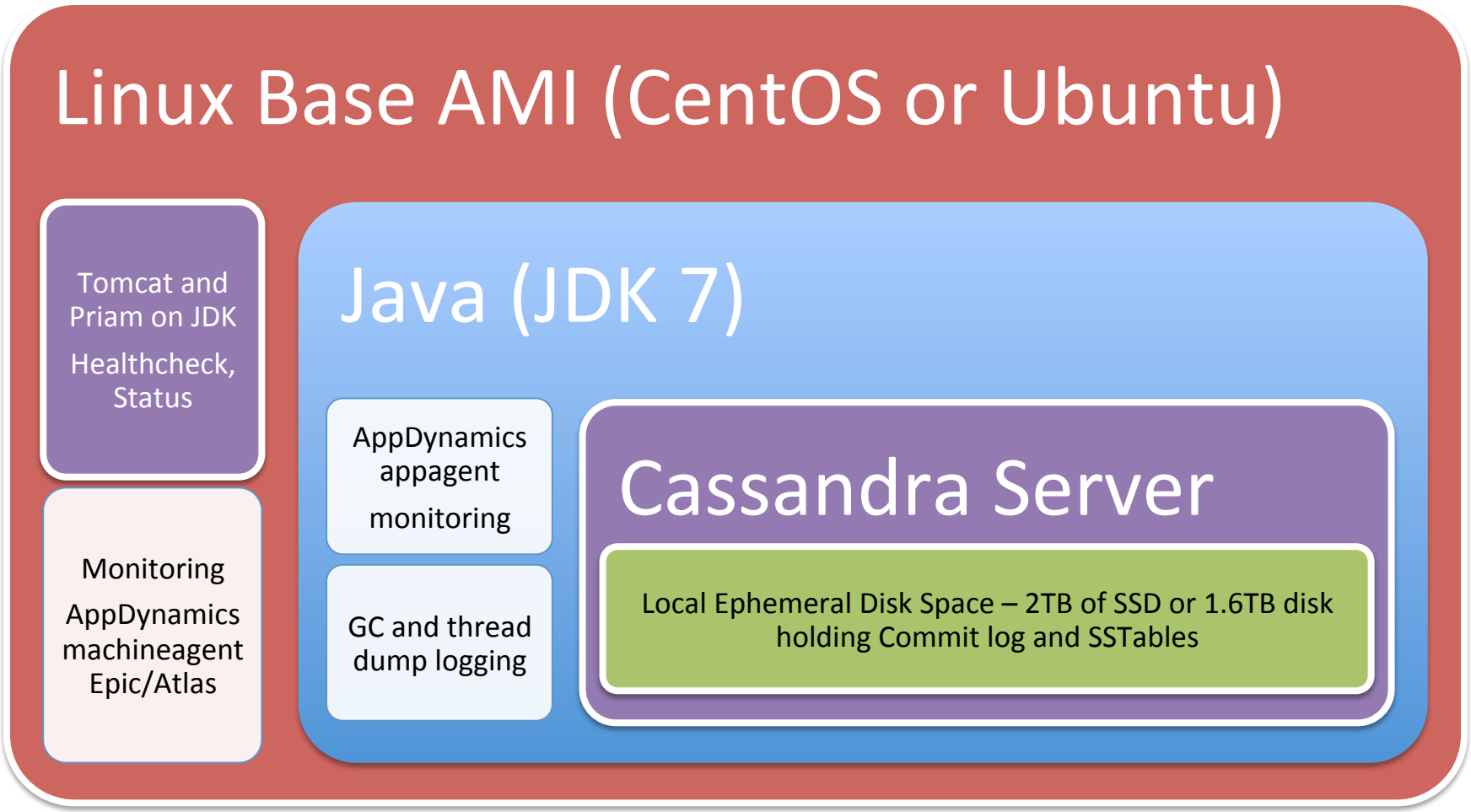
1. Client writes to local replicas
2. Local write acks returned to Client which continues when 2 of 3 local nodes are committed
3. Local coordinator writes to remote coordinator.
4. When data arrives, remote coordinator node acks and copies to other remote zones
5. Remote nodes ack to local coordinator
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.



NETFLIX

Cassandra Instance Architecture



NETFLIX

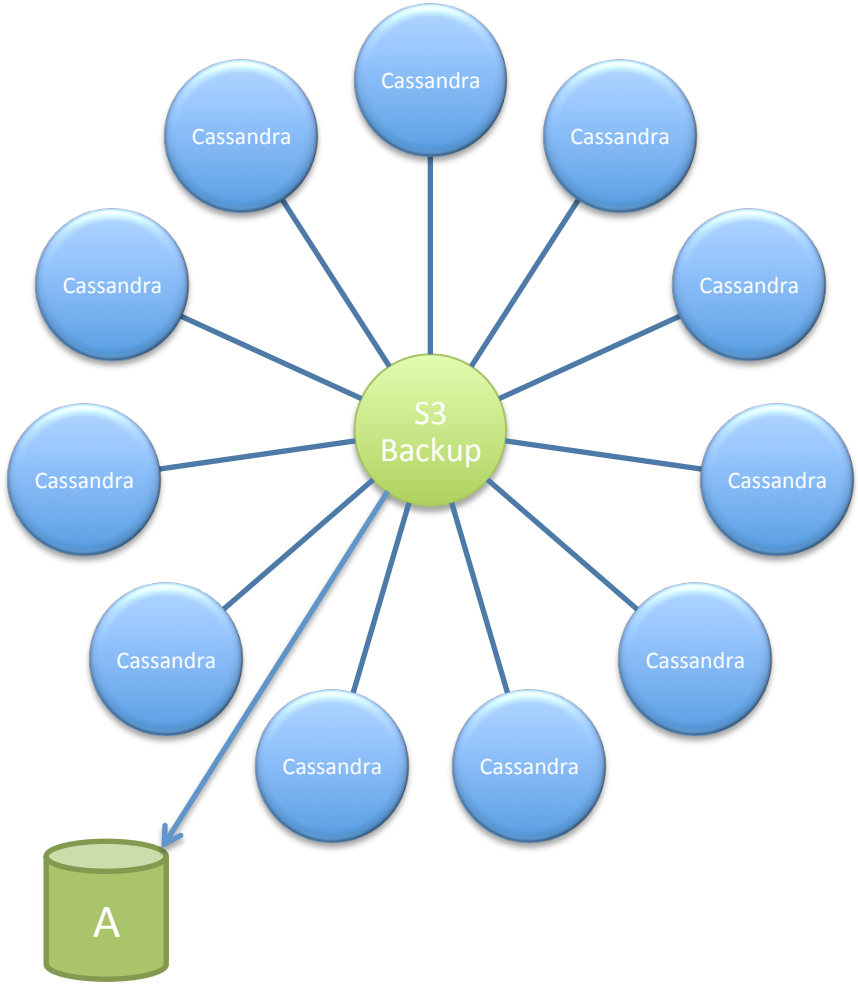
Priam – Cassandra Automation

Available at http://github.com/netflix

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

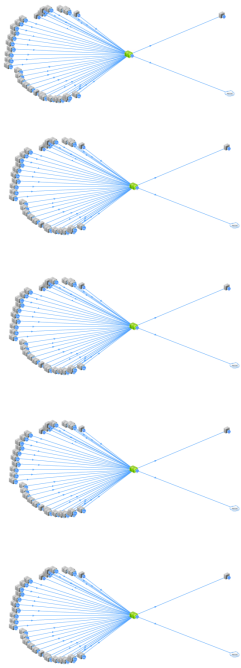
Cassandra Backup

- Full Backup
  - Time based snapshot
  - SSTable compress -> S3
- Incremental
  - SSTable write triggers compressed copy to S3
- Archive
  - Copy cross region



NETFLIX

Production Deployment

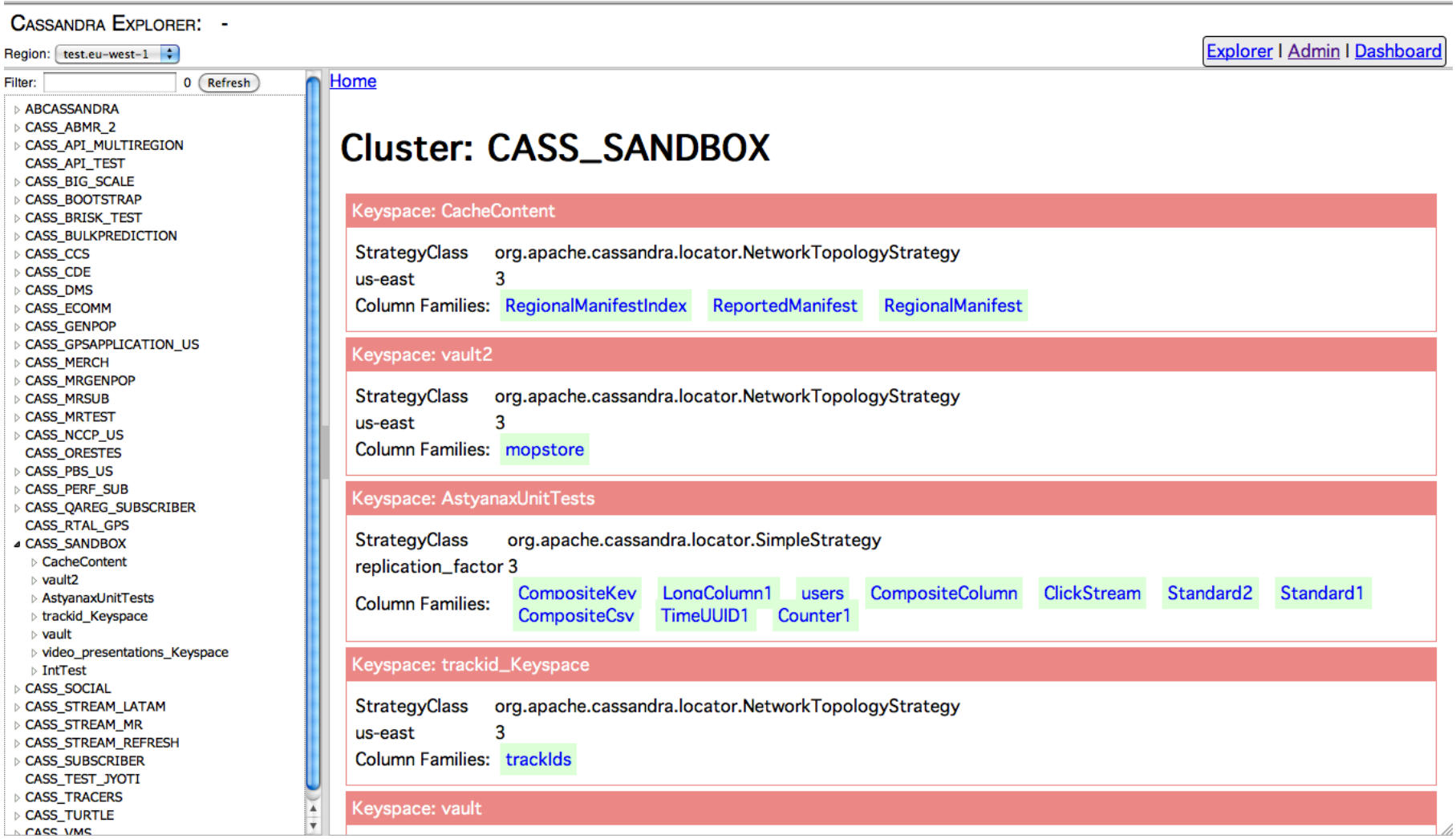


- Over 50 Cassandra Clusters
- Over 500 nodes
- Over 30TB of daily backups
- Biggest cluster 72 nodes
- 1 cluster over 250Kwrites/s

NETFLIX

Cassandra Explorer for Data

Open source on github soon



@adrianco

NETFLIX

ETL for Cassandra

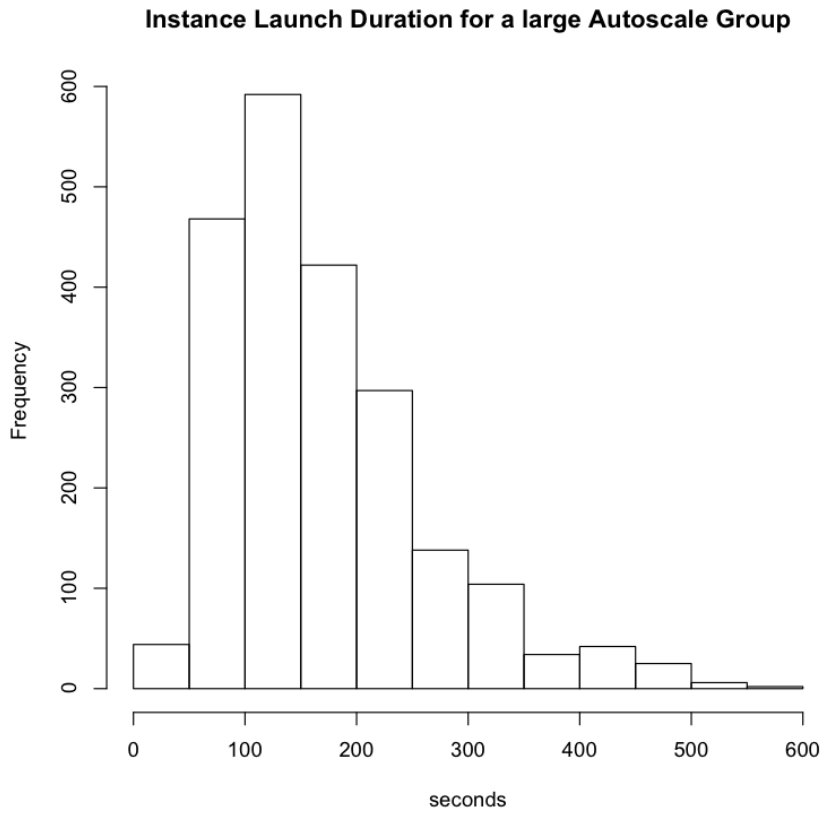
- Data is de-normalized over many clusters!
- Too many to restore from backups for ETL
- Solution – read backup files using Hadoop
- Aegisthus
  - http://techblog.netflix.com/2012/02/aegisthus-bulk-data-pipeline-out-of.html
  - High throughput raw SSTable processing
  - Re-normalizes many clusters to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

Benchmarks and Scalability

Cloud Deployment Scalability

New Autoscaled AMI – zero to 500 instances from 21:38:52 - 21:46:32, 7m40s
Scaled up and down over a few days, total 2176 instance launches, m2.2xlarge (4 core 34GB)

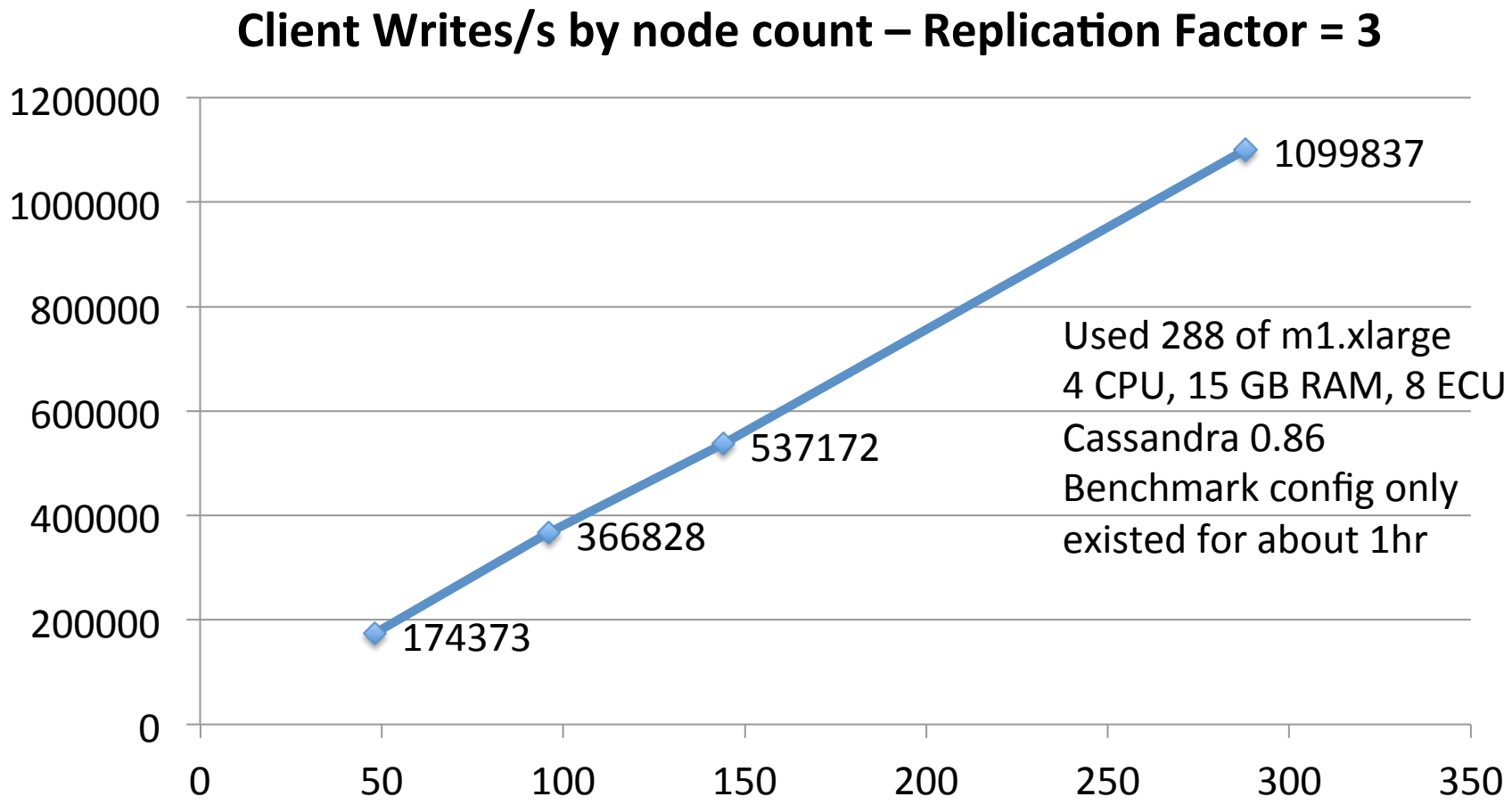
```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
41.0   104.2   149.0   171.8   215.8   562.0
```



NETFLIX

Scalability from 48 to 288 nodes on AWS

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



“Some people skate to the puck,
I skate to where the puck is going to be”
Wayne Gretzky

NETFLIX

Cassandra on AWS

The Past

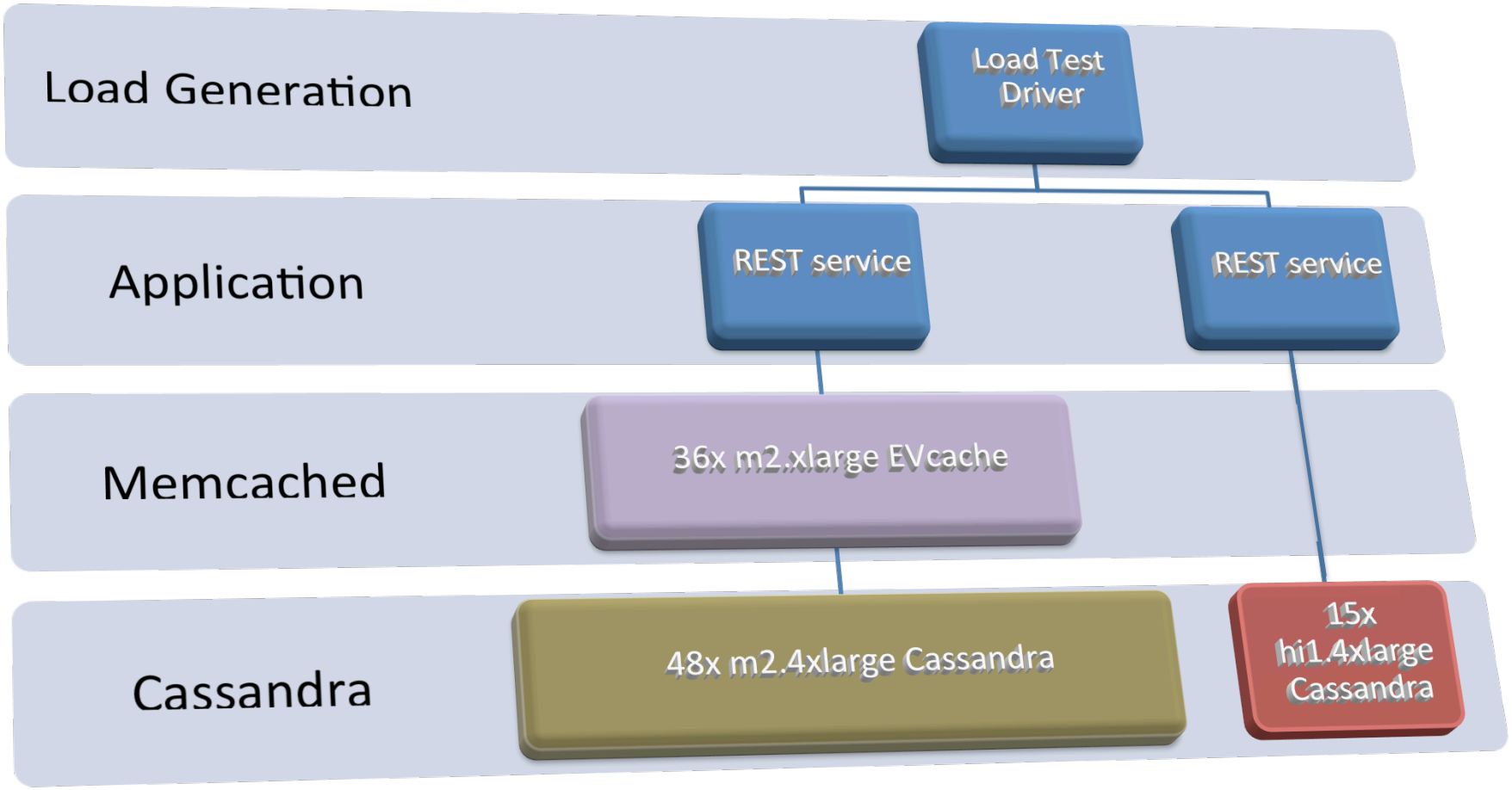
- Instance: m2.4xlarge
- Storage: 2 drives, 1.7TB
- CPU: 8 Cores, 26 ECU
- RAM: 68GB
- Network: 1Gbit
- IOPS: ~500
- Throughput: ~100Mbyte/s
- Cost: $1.80/hr

The Future

- Instance: hi1.4xlarge
- Storage: 2 SSD volumes, 2TB
- CPU: 8 HT cores, 35 ECU
- RAM: 64GB
- Network: 10Gbit
- IOPS: ~100,000
- Throughput: ~1Gbyte/s
- Cost: $3.10/hr

Cassandra Disk vs. SSD Benchmark

Same Throughput, Lower Latency, Half Cost



NETFLIX

Netflix Open Source Strategy

- Release PaaS Components git-by-git
  - Source at github.com/netflix – we build from it…
  - Intros and techniques at techblog.netflix.com
  - Blog post or new code every few weeks

- Motivations
  - Give back to Apache licensed OSS community
  - Motivate, retain, hire top engineers
  - "Peer pressure" code cleanup, external contributions

Instance creation

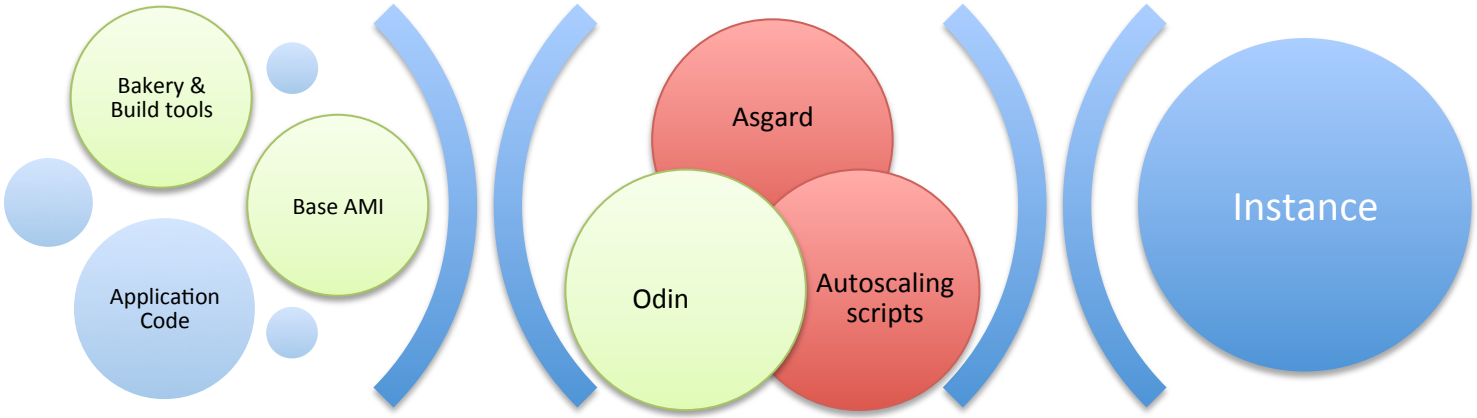
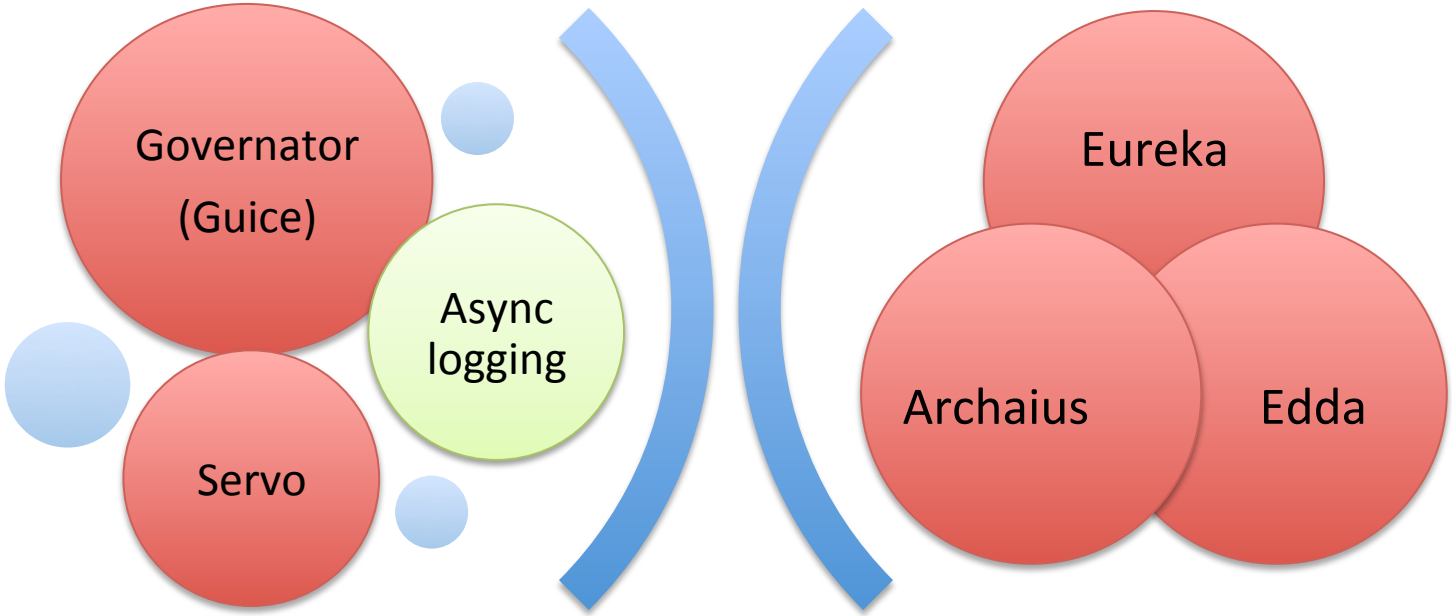


Image baked

ASG / Instance started

Instance Running

Application Launch

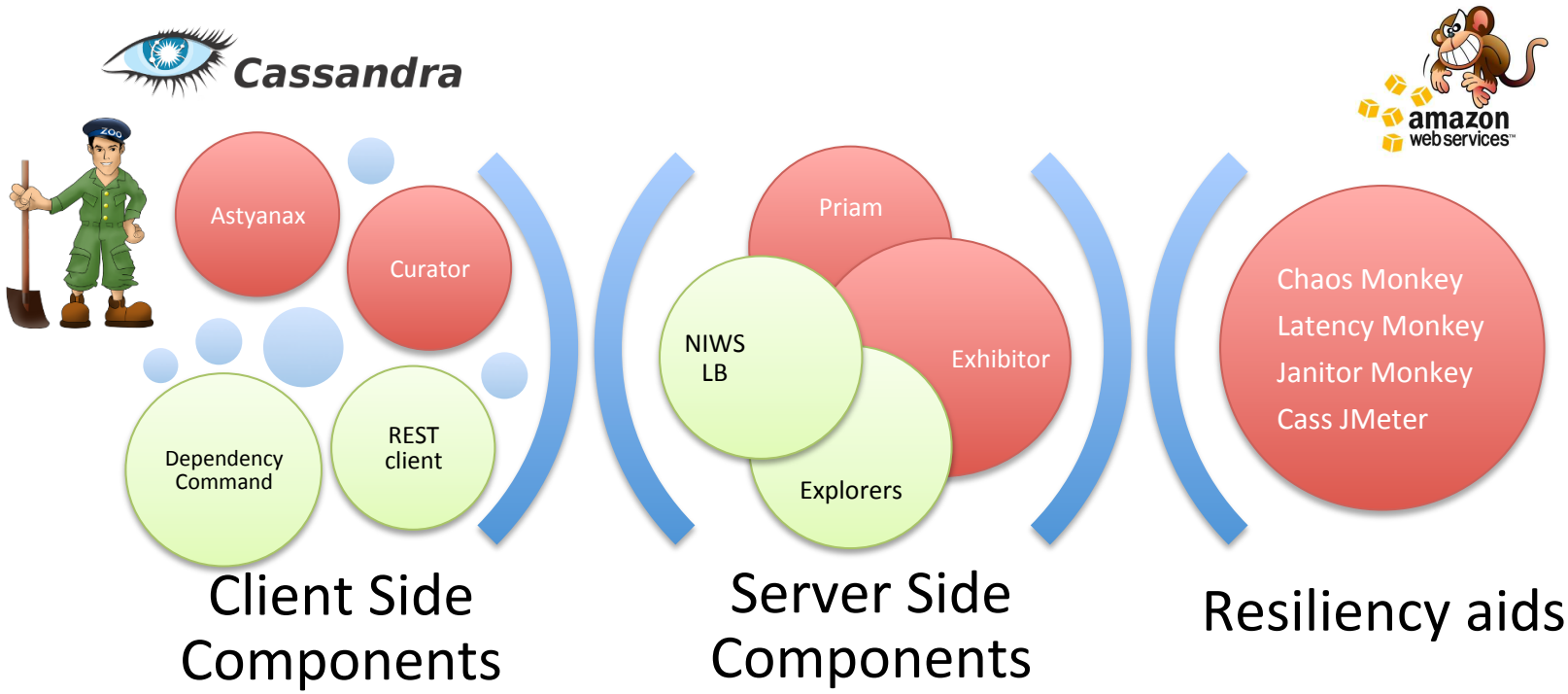


Application initializing

Service Registry,
configuration history

NETFLIX

Runtime



NETFLIX

Open Source Projects

Legend

- Github / Techblog
- Apache Contributions
- Techblog Post
- Coming Soon

Priam Cassandra as a Service	Exhibitor Zookeeper as a Service	Servo and Autoscaling Scripts
Astyanax Cassandra client for Java	Curator Zookeeper Patterns	Honu Log4j streaming to Hadoop
CassJMeter Cassandra test suite	EVCache Memcached as a Service	Circuit Breaker Robust service pattern
Cassandra Multi-region EC2 datastore support	Eureka / Discovery Service Directory	Asgard - AutoScaleGroup based AWS console
Aegisthus Hadoop ETL for Cassandra	Archaius Dynamics Properties Service	Chaos Monkey Robustness verification
Explorers	Edda Queryable config history	Latency Monkey
Governator - Library lifecycle and dependency injection	Server-side latency/error injection	Janitor Monkey
Odin Workflow orchestration	REST Client + mid-tier LB	Bakeries and AMI
Async logging	Configuration REST endpoints	Build dynaslaves

@adrianco

NETFLIX

Cassandra Next Steps

- Migrate Production Cassandra to SSD
  - Many clusters done
  - 100+ SSD nodes running
- Autoscale Cassandra using Priam
  - Cassandra 1.2 Vnodes make this easier
  - Shrink Cassandra cluster every night
- Automated Zone and Region Operations
  - Add/Remove Zone, split or merge clusters
  - Add/Remove Region, split or merge clusters

Skynet

A Netflix Hackday project that might just terminate the world…

(hack currently only implemented in Powerpoint – luckly)

@adrianco

The Plot (kinda)

- Skynet is a sentient computer
- Skynet defends itself if you try to turn it off
- Connor is the guy who eventually turns it off
- Terminator is the robot sent to kill Connor

The Hacktors

- Cass_skynet is a self-managing Cassandra cluster
- Connor_monkey kills cass_skynet nodes
- Terminator_monkey kills connor_monkey nodes

NETFLIX

The Hacktion

- Cass_skynet stores a history of its world and action scripts that trigger from what it sees
- Action response to losing a node
  - Auto-replace node and grow cluster size
- Action response to losing more nodes
  - Replicate cluster into a new zone or region
- Action response to seeing a Connor_monkey
  - Startup a Terminator_monkey

Implementation

- Priam
  - Autoreplace missing nodes
  - Grow cass_skynet cluster in zone, to new zones or regions
- Cassandra Keyspaces
  - Actions – scripts to be run
  - Memory – record event log of everything seen
- Cron job once a minute
  - Extract actions from Cassandra and execute
  - Log actions and results in memory
- Chaos Monkey configuration
  - Terminator_monkey: pick a zone, kill any connor_monkey
  - Connor_monkey: kill any cass_skynet or terminator_monkey

“Simulation”


TERMINATOR
SALVATION


SKYNET
NEURAL NET-BASED ARTIFICIAL INTELLIGENCE


SKYNET

High Anxiety

Takeaway

Netflix has built and deployed a scalable global platform based on Cassandra and AWS.

Key components of the Netflix PaaS are being released as Open Source projects so you can build your own custom PaaS.

SSD's in the cloud are awesome….

http://github.com/Netflix
http://techblog.netflix.com
http://slideshare.net/Netflix

http://www.linkedin.com/in/adriancockcroft
@adrianco http://perfcap.blogspot.com

NETFLIX