Netflix Cloud Architecture

Qcon Beijing April 9, 2011

Adrian Cockcroft

@adrianco #netflixcloud http://slideshare.net/adrianco

acockcroft@netflix.com

(Continuing from Keynote Talk)

~~Who, Why~~, What

~~Netflix in the Cloud~~

~~Cloud Challenges and Learnings~~

Systems and Operations Architecture

Amazon Cloud Terminology

See http://aws.amazon.com/ for details
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)
- RDB – Relational Data Base (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)

amazon web services™

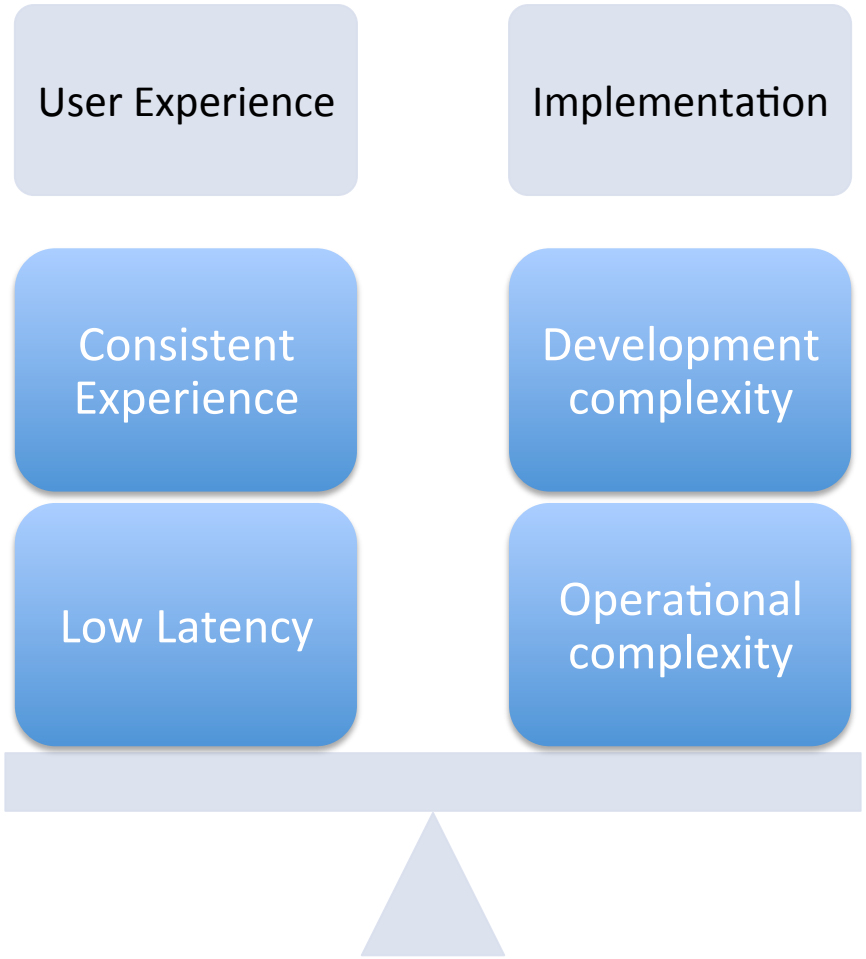
NETFLIX

Netflix Deployed on AWS

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

Cloud Architecture

Product Trade-off



NETFLIX

Synopsis

- The Goals
  - Faster, Scalable, Available and Productive
- Anti-patterns and Cloud Architecture
  - The things we wanted to change and why
- Capacity Planning and Monitoring
- Next Steps

Netflix Cloud 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

NETFLIX

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

NETFLIX

The Central SQL Database

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

This Anti-pattern impacts scalability, availability

NETFLIX

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 Memcached vs. Oracle vs. SimpleDB
  - Memcached is dominated by network latency <1ms
  - Oracle for simple queries is a few milliseconds
  - SimpleDB has replication and REST overheads >10ms

NETFLIX

Database Migration

- Why SimpleDB?
  - No DBA's in the cloud, Amazon hosted service
  - Work started two years ago, fewer viable options
  - Worked with Amazon to speed up and scale SimpleDB
- Alternatives?
  - Now rolling out Cassandra as "upgrade" from SimpleDB
  - Need several options to match use cases well
- Detailed NoSQL and SimpleDB Advice
  - Sid Anand - QConSF Nov 5th – Netflix' Transition to High Availability Storage Systems
  - Blog - http://practicalcloudcomputing.com/
  - Download Paper PDF - http://bit.ly/bh0TLu

NETFLIX

Oracle to SimpleDB

(See Sid's paper for details)

- SimpleDB Domains
  - De-normalize multiple tables into a single domain
  - Work around size limits (10GB per domain, 1KB per key)
  - Shard data across domains to scale
  - Key – Use distributed sequence generator, GUID or natural unique key such as customer-id
  - Implement a schema validator to catch bad attributes
- Application layer support
  - Do GROUP BY and JOIN operations in the application
  - Compose relations in the application layer
  - Check constraints on read, and repair data as a side effect
- Do without triggers, PL/SQL, clock operations

The Sticky Session

- Datacenter Sticky Load Balancing
  - Efficient caching for low latency
  - Tricky session handling code
  - Middle tier load balancer has issues in practice
- Encourages concentrated functionality
  - one service that does everything

This Anti-pattern impacts productivity, availability

NETFLIX

The Shared Session

- Cloud Uses Round-Robin Load Balancing
  - Simple request-based code
  - External shared caching with memcached
- More flexible fine grain services
  - Works better with auto-scaled instance counts

NETFLIX

Chatty Opaque and Brittle Protocols

- Datacenter service protocols
  - Assumed low latency for many simple requests
- Based on serializing existing java objects
  - Inefficient formats
  - Incompatible when definitions change

This Anti-pattern causes productivity, latency and availability issues

NETFLIX

Robust and Flexible Protocols

- Cloud service protocols
  - JSR311/Jersey is used for REST/HTTP service calls
  - Custom client code includes service discovery
  - Support complex data types in a single request
- Apache Avro
  - Evolved from Protocol Buffers and Thrift
  - Includes JSON header defining key/value protocol
  - Avro serialization is **half the size** and several times faster than Java serialization, more work to code

Persisted Protocols

- Persist Avro in Memcached
  - Save space/latency (zigzag encoding, half the size)
  - Less brittle across versions
  - New keys are ignored
  - Missing keys are handled cleanly
- Avro protocol definitions
  - Can be written in JSON or generated from POJOs
  - It's hard, needs better tooling

NETFLIX

Tangled Service Interfaces

- Datacenter implementation is exposed
  - Oracle SQL queries mixed into business logic
- Tangled code
  - Deep dependencies, false sharing
- Data providers with sideways dependencies
  - Everything depends on everything else

This Anti-pattern affects productivity, availability

Untangled Service Interfaces

- New Cloud Code With Strict Layering
  - Compile against interface jar
  - Can use spring runtime binding to enforce

- Service interface **is** the service
  - Implementation is completely hidden
  - Can be implemented locally or remotely
  - Implementation can evolve independently

Untangled Service Interfaces

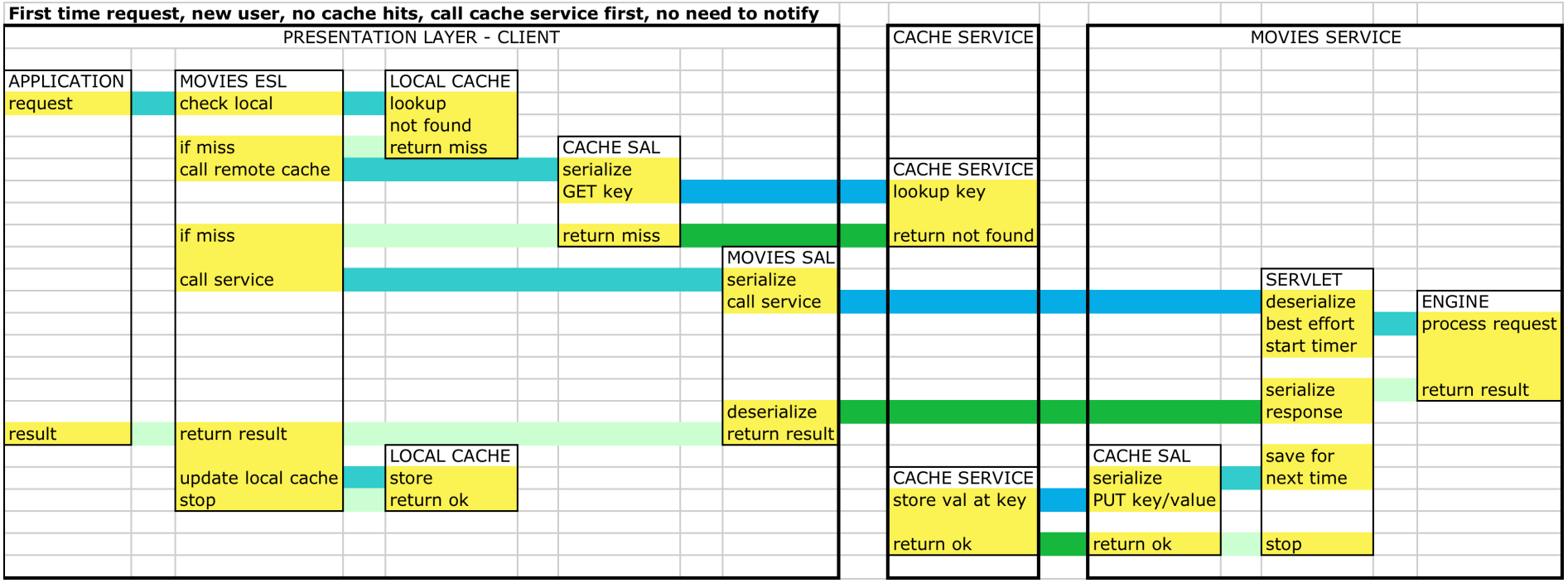
Two layers:

- SAL - Service Access Library
  - Basic serialization and error handling
  - REST or POJO's defined by data provider

- ESL - Extended Service Library
  - Caching, conveniences
  - Can combine several SALs
  - Exposes faceted type system (described later)
  - Interface defined by data consumer in many cases

Service Interaction Pattern

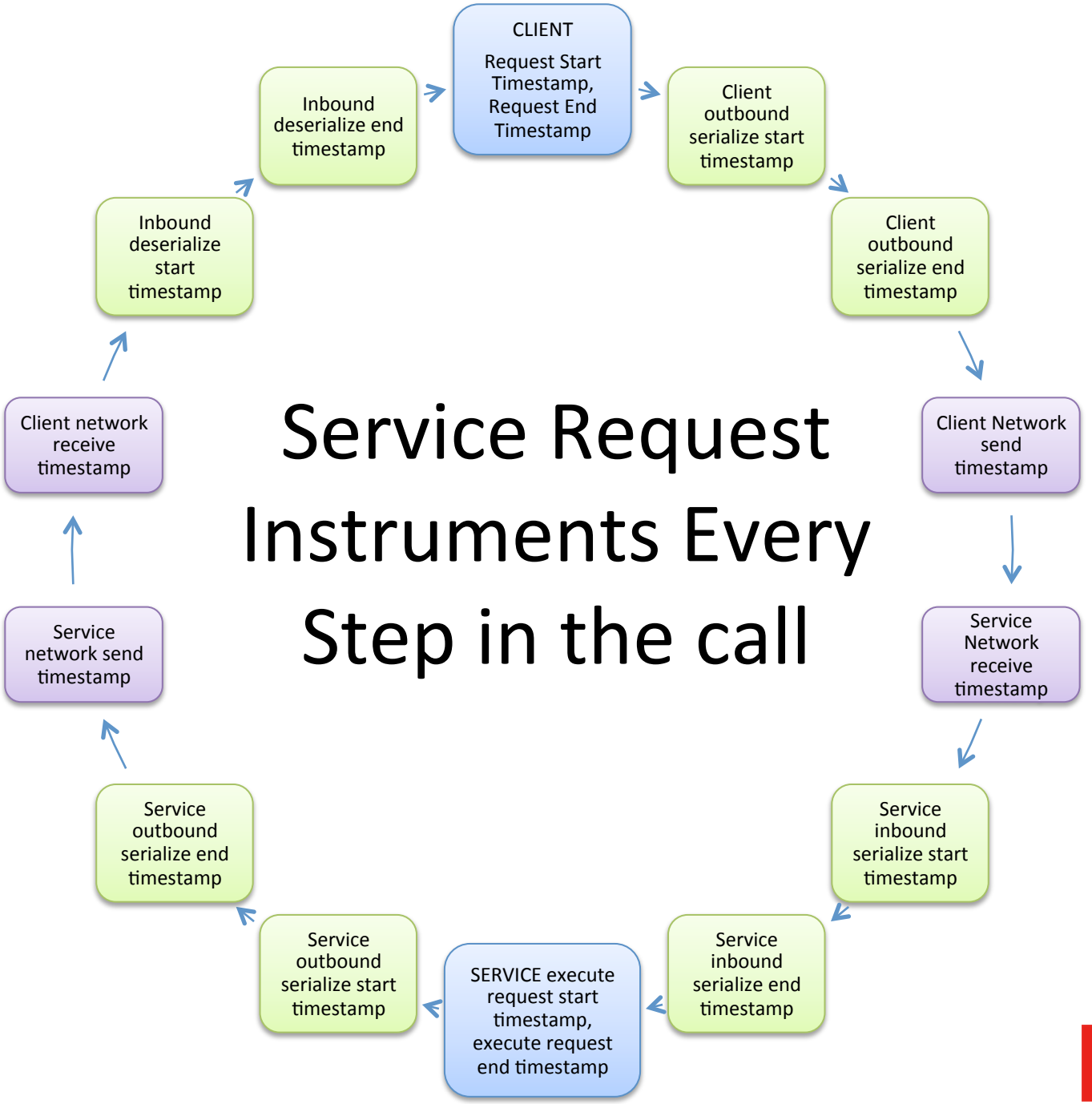
Sample Swimlane Diagram



Service Architecture Patterns

- Internal Interfaces Between Services
  - Common patterns as templates
  - Highly instrumented, observable, analytics
  - Service Level Agreements – SLAs
- Library templates for generic features
  - Instrumented Netflix Base Servlet template
  - Instrumented generic client interface template
  - Instrumented S3, SimpleDB, Memcached clients

NETFLIX


CLIENT
Request Start Timestamp, Request End Timestamp
Client outbound serialize start timestamp
Client outbound serialize end timestamp
Client Network send timestamp
Service Network receive timestamp
Service inbound serialize start timestamp
Service inbound serialize end timestamp
SERVICE execute request start timestamp, execute request end timestamp
Service outbound serialize start timestamp
Service outbound serialize end timestamp
Service network send timestamp
Client network receive timestamp
Inbound deserialize start timestamp
Inbound deserialize end timestamp
Service Request Instruments Every Step in the call

NETFLIX

Boundary Interfaces

- Isolate teams from external dependencies
  - Fake SAL built by cloud team
  - Real SAL provided by data provider team later
  - ESL built by cloud team using faceted objects
- Fake data sources allow development to start
  - e.g. Fake Identity SAL for a test set of customers
  - Development solidifies dependencies early
  - Helps external team provide the right interface

NETFLIX

One Object That Does Everything

- Datacenter uses a few big complex objects
  - Movie and Customer objects are the foundation
  - Good choice for a small team and one instance
  - Problematic for large teams and many instances
- False sharing causes tangled dependencies
  - Unproductive re-integration work

Anti-pattern impacting productivity and availability

An Interface For Each Component

- Cloud uses faceted Video and Visitor
  - Basic types hold only the identifier
  - Facets scope the interface you actually need
  - Each component can define its own facets
- No false-sharing and dependency chains
  - Type manager converts between facets as needed
  - video.asA(PresentationVideo) for www
  - video.asA(MerchableVideo) for middle tier

Software Architecture Patterns

- Object Models
  - Basic and derived types, facets, serializable
  - Pass by reference within a service
  - Pass by value between services
- Computation and I/O Models
  - Service Execution using Best Effort
  - Common thread pool management

NETFLIX

Cloud Operations

Model Driven Architecture

Capacity Planning & Monitoring

Tools and Automation

- Developer and Build Tools
  - Jira, Eclipse, Jeeves, 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/SunJDK, CentOS/AmazonLinux
- Monitoring Tools
  - Keynote – service monitoring and alerting
  - AppDynamics – Developer focus for cloud http://appdynamics.com
  - EpicNMS – flexible data collection and plots http://epicnms.com
  - Nimsoft NMS – ITOps focus for Datacenter + Cloud alerting

NETFLIX

Model Driven Architecture

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

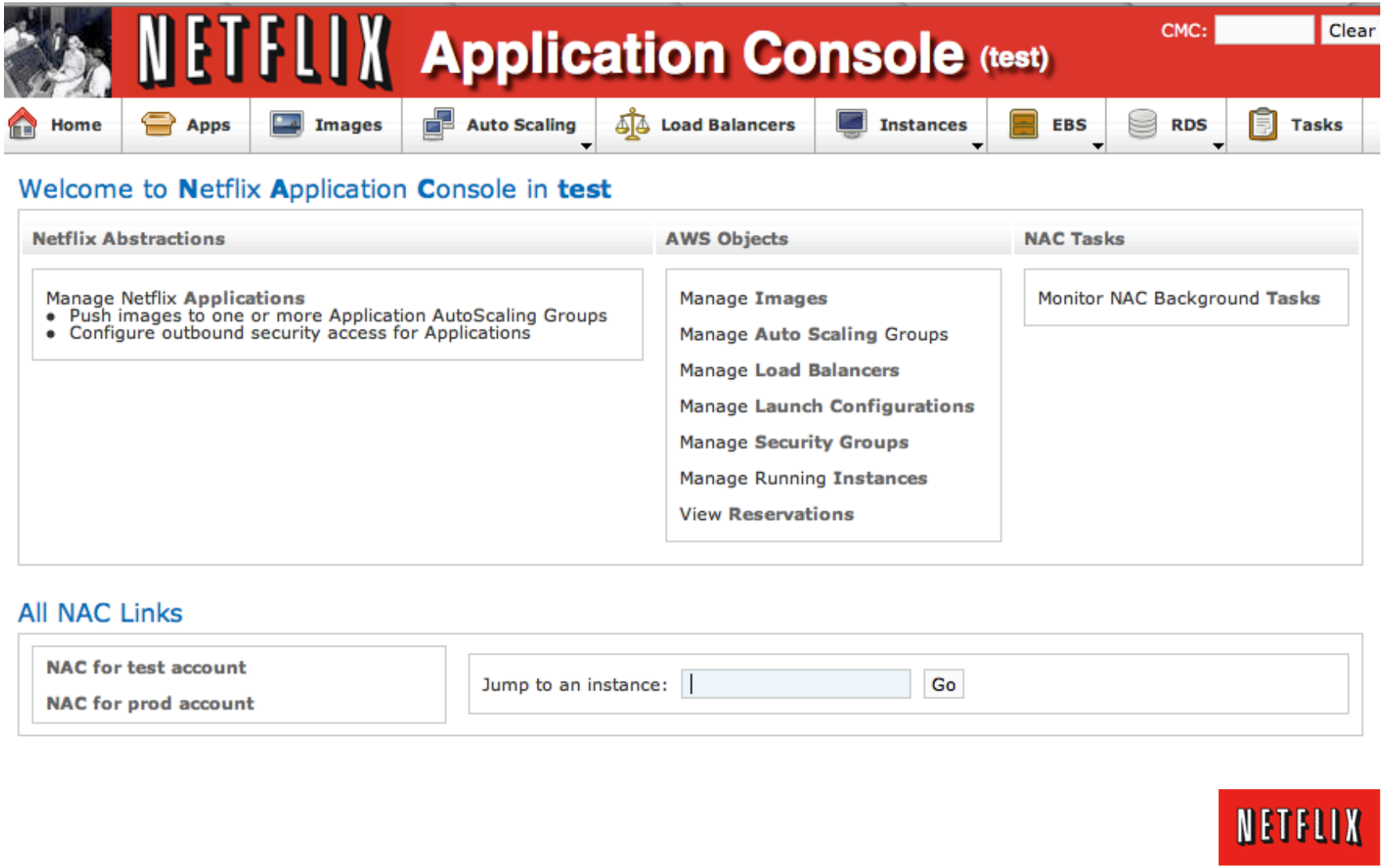
No exceptions, every change is a new AMI

Model Driven Implications

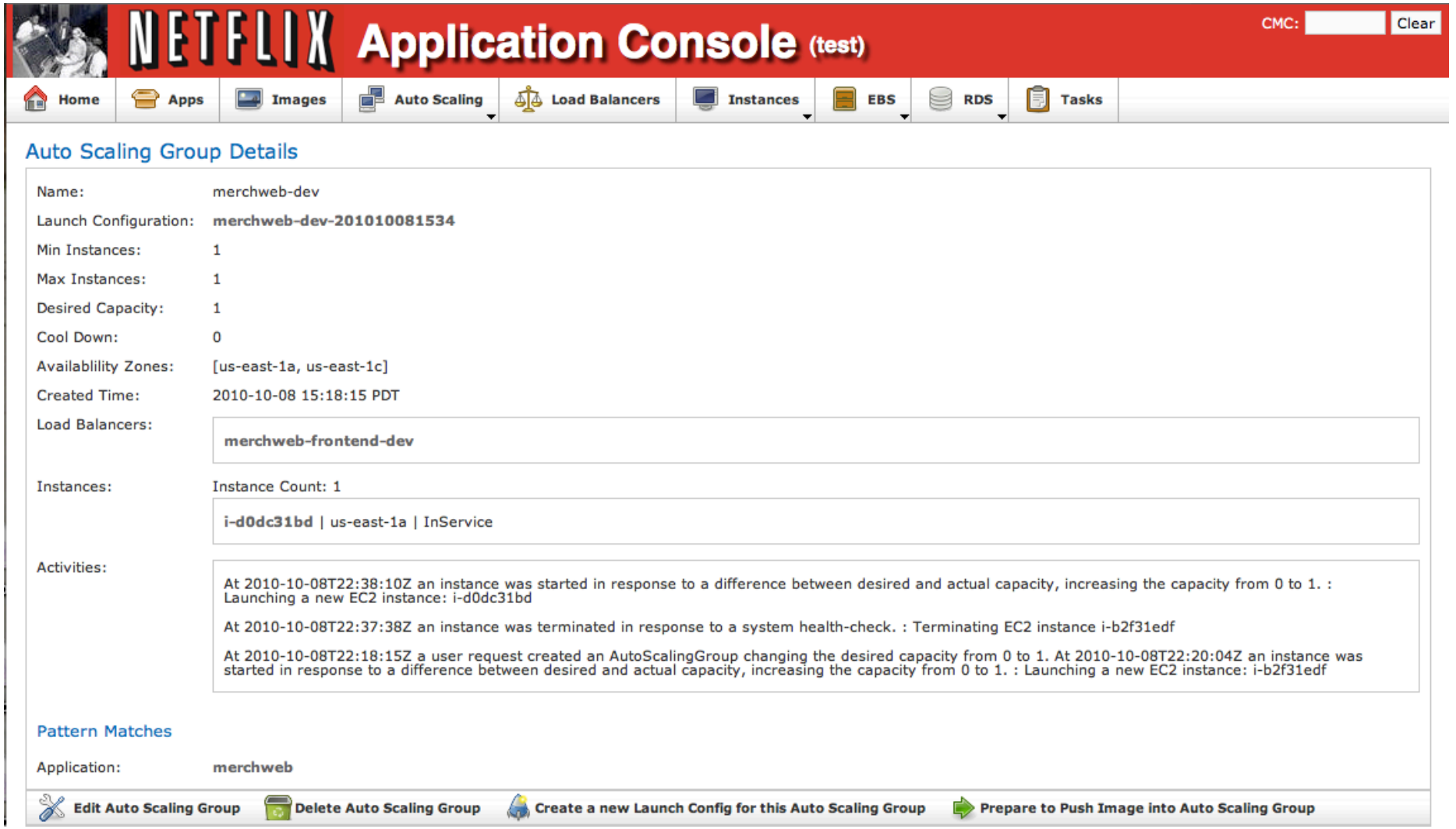
- Automated “Least Privilege” Security
  - Tightly specified security groups
  - Fine grain IAM keys to access AWS resources
  - Performance tools security and integration
- Model Driven Performance Monitoring
  - Hundreds of instances appear in a few minutes…
  - Tools have to “garbage collect” dead instances

NETFLIX

Netflix App Console



Auto Scale Group Configuration



NETFLIX

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!)

NETFLIX

Monitoring Issues

- Problem
  - Too many tools, each with a good reason to exist
  - Hard to get an integrated view of a problem
  - Too much manual work building dashboards
  - Tools are not discoverable, views are not filtered
- Solution
  - Get vendors to add deep linking URLs and APIs
  - Integration "portal" ties everything together
  - Underlying dependency database
  - Dynamic portal generation, relevant data, all tools

NETFLIX

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 – Nimsoft, Appdynamics, Epic •Service and SLA percentiles – Nimsoft, Appdynamics, Epic,logged to DataOven
Tomcat and Apache logs	•Stdout logs – S3 – DataOven, Nimsoft alerting •Standard format Access and Error logs – S3 – DataOven, Nimsoft Alerting
JVM	•Garbage Collection – Nimsoft, Appdynamics •Memory usage, call stacks, resource/call - AppDynamics
Linux	•system CPU/Net/RAM/Disk metrics – AppDynamics, Epic, Nimsoft Alerting •SNMP metrics – Epic, Network flows - Fastip
AWS	•Load balancer traffic – Amazon Cloudwatch, SimpleDB usage stats •System configuration - CPU count/speed and RAM size, overall usage - AWS

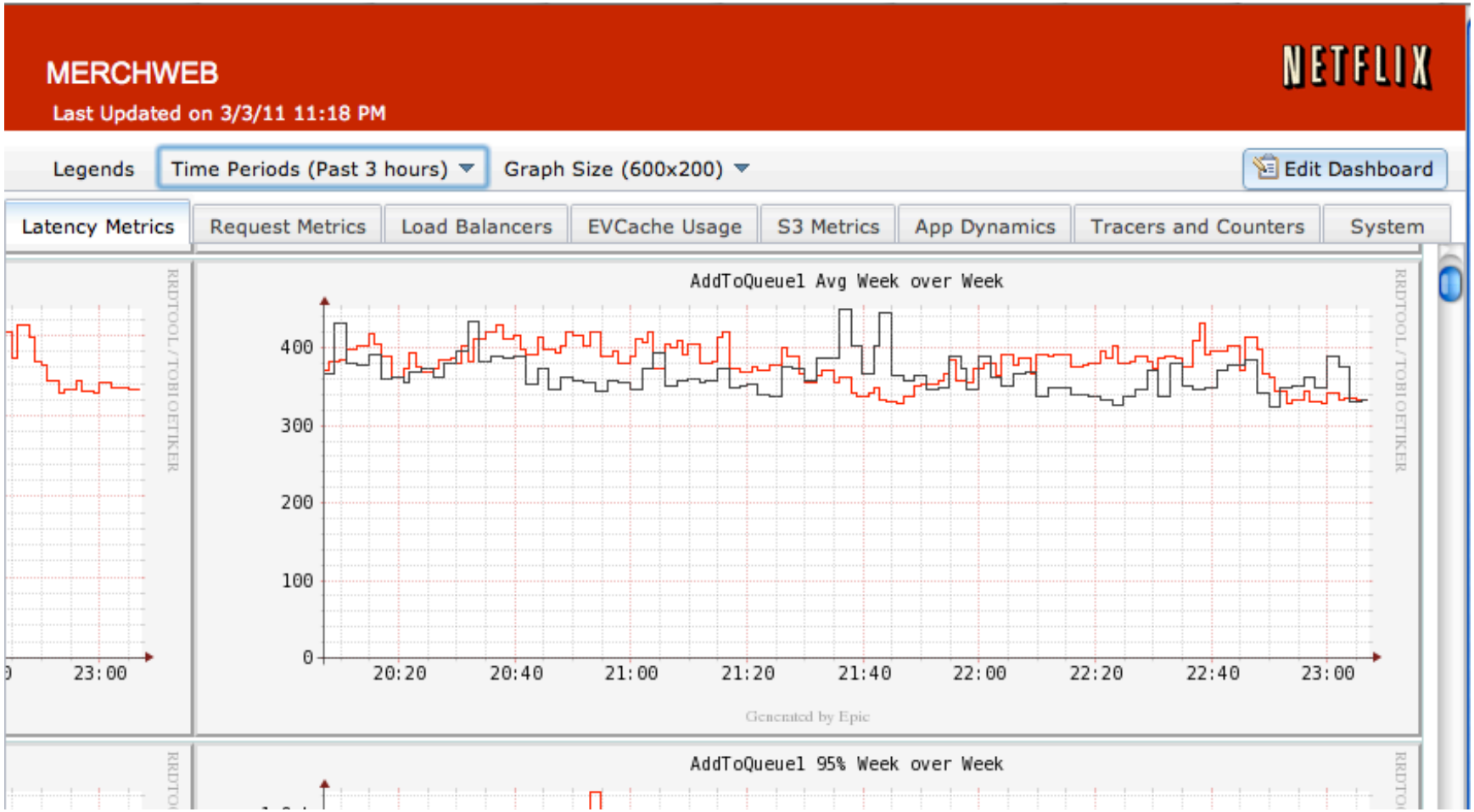
Integrated Dashboards

Dashboards Architecture

- Integrated Dashboard View
  - Single web page containing content from many tools
  - Filtered to highlight most “interesting” data
- Relevance Controller
  - Drill in, add and remove content interactively
  - Given an application, alert or problem area, dynamically build a dashboard relevant to your role and needs
- Dependency and Incident Model
  - Model Driven - Interrogates tools and AWS APIs
  - Document store to capture dependency tree and states

Dashboard Prototype

(not everything is integrated yet)



NETFLIX

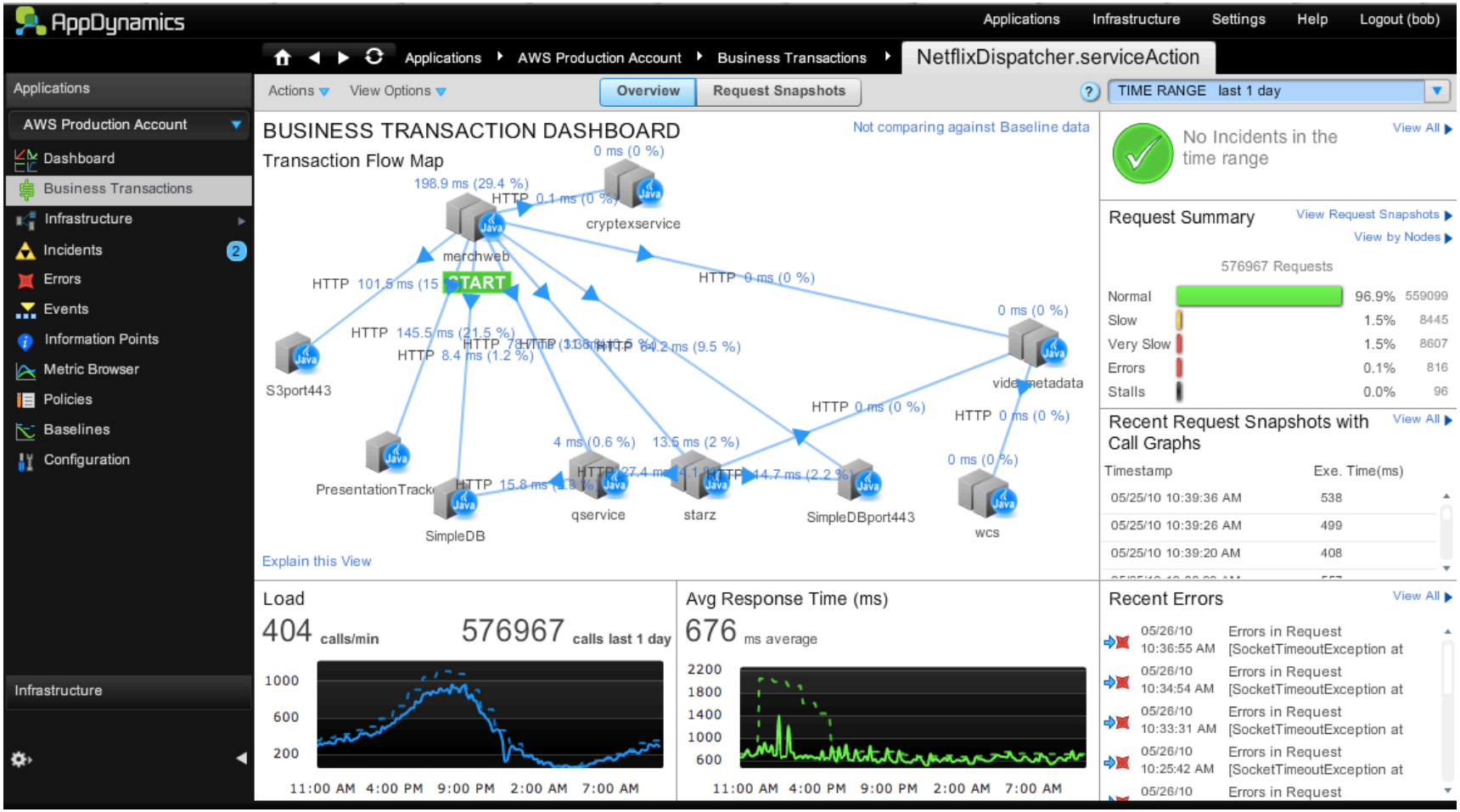
AppDynamics

How to look deep inside your cloud applications

- Automatic Monitoring
  - Base AMI includes 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

Using AppDynamics

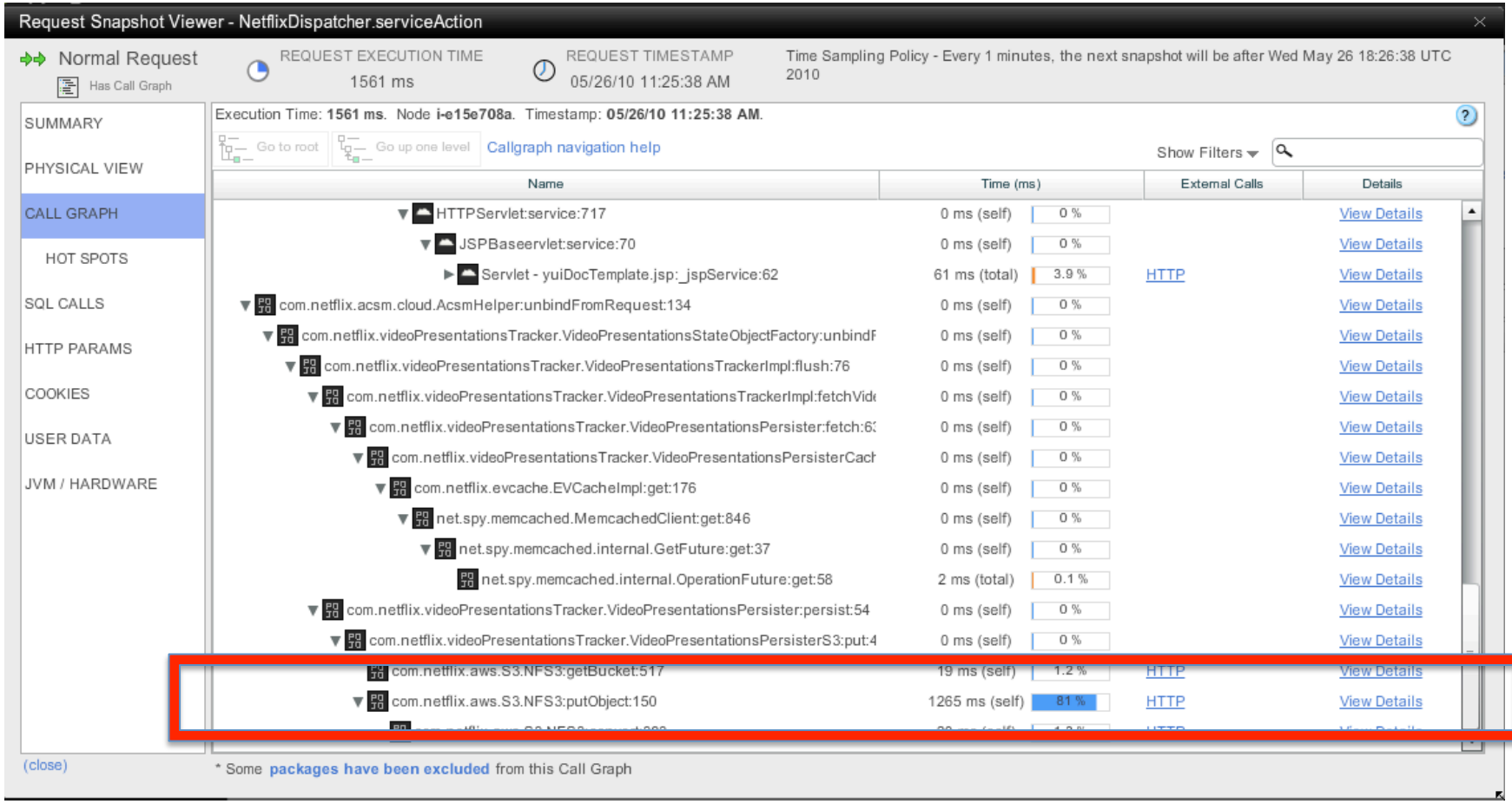
(simple example from early 2010)



NETFLIX

Assess Impact using AppDynamics

View actual call graph on production systems



NETFLIX

Monitoring Summary

- Broken datacenter oriented tools is a big problem
- Integrating many different tools
  - They are not designed to be integrated
  - We have “persuaded” vendors to add APIs
- If you can’t see deep inside your app, you’re ☹

Wrap Up

Next Few Years…

- "System of Record" moves to Cloud (now)
  - Master copies of data live only in the cloud, with backups
  - Cut the datacenter to cloud replication link
- International Expansion – Global Clouds (later in 2011)
  - Rapid deployments to new markets
- Cloud Standardization?
  - Cloud features and APIs should be a commodity not a differentiator
  - Differentiate on scale and quality of service
  - Competition also drives cost down
  - Higher resilience and scalability

We would prefer to be an insignificant customer in a giant cloud

NETFLIX

Takeaway

Netflix is path-finding the use of public AWS cloud to replace in-house IT for non-trivial applications with hundreds of developers and thousands of systems.

acockcroft@netflix.com

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

@adrianco #netflixcloud

amazon web services™

NETFLIX