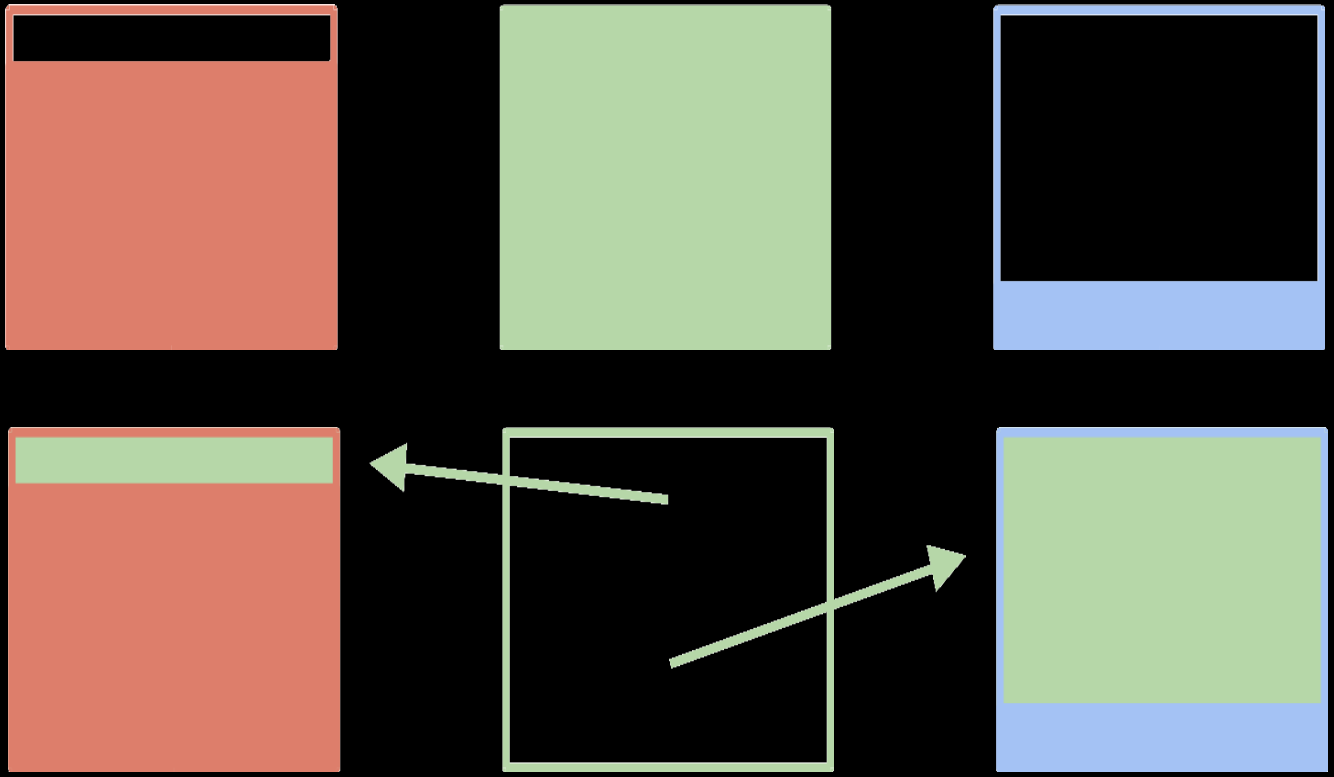
NETFLIX

Chaos Kong
Endowing Netflix with Antifragility

Luke Kosewski Traffic & Chaos Engineering

This is a Case Study We’ll Be Doing TOGETHER

This is What AWS Failover Looks Like



Failover is Run By This Guy

A Traffic Engineer

Failover is Run By This Guy

A Traffic Engineer

A Traffic Engineer’s Environment

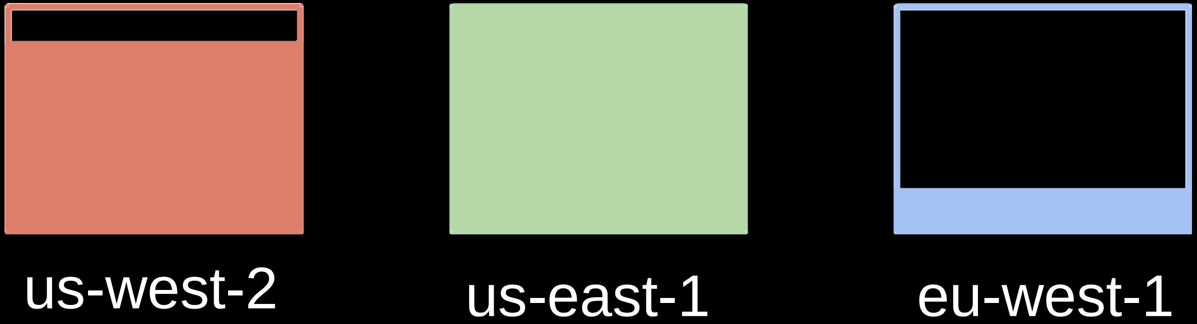
- Netflix control plane

A Traffic Engineer’s Environment

- Netflix control plane
- Primarily in 3 AWS regions (EU, us-east-1, us-west-2)

A Traffic Engineer’s Environment

- Netflix control plane
- Primarily in 3 AWS regions (EU, us-east-1, us-west-2)
- They look like this:



us-west-2

us-east-1

eu-west-1

Traffic’s Teammates

traffic@netflix.com / chaos@netflix.com

Traffic

Niosha Behnam & myself

Intuition

Justin Reynolds

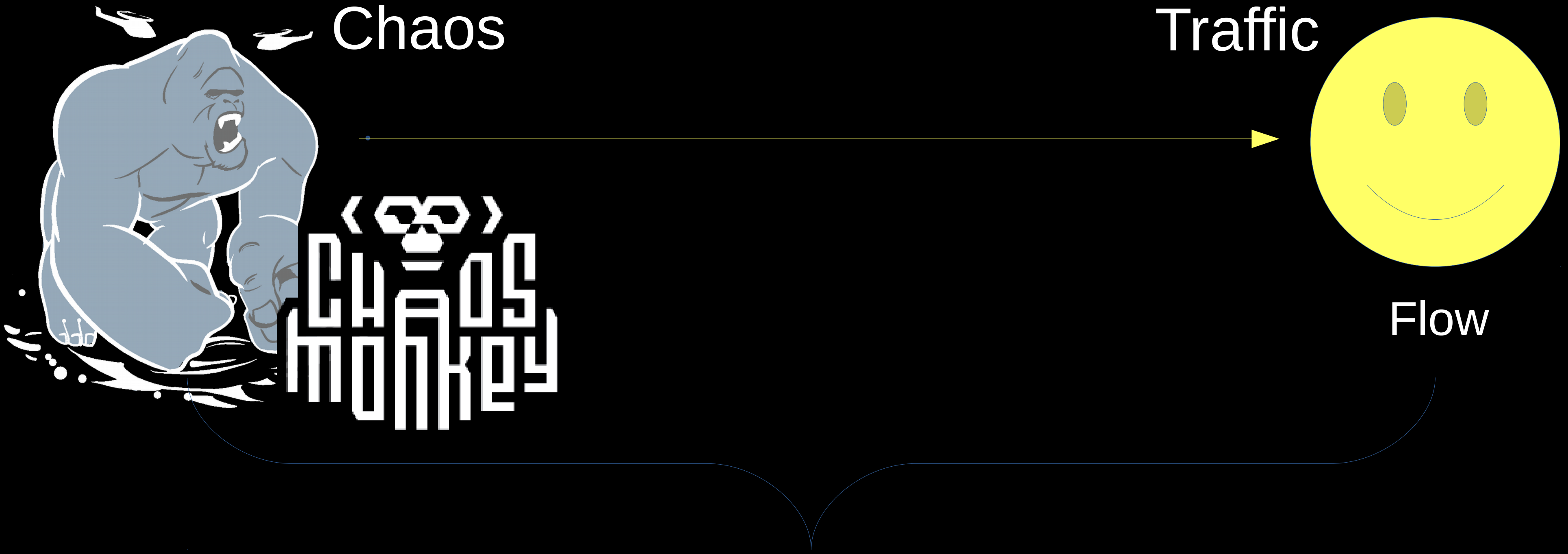
Chaos

Lorin Hochstein, Aaron Blohowiak & Ali Basiri

(management)

Casey Rosenthal

Our Relationship



Chaos

Traffic

Flow

High Availability

Storytime with Luke

Once upon a time...
(August 2013)

3 SREs at Netflix

3 SREs at Netflix

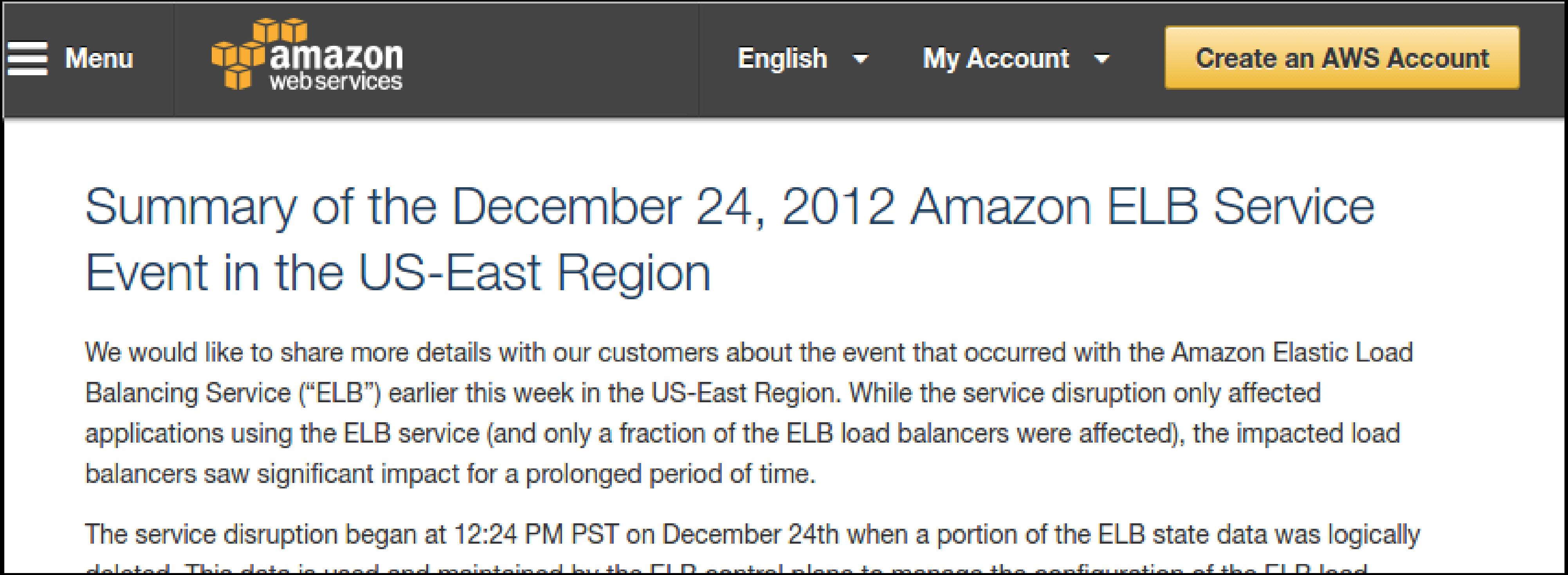
10s of services

3 SREs at Netflix

10s of services

100s of devs

Disaster



Summary of the December 24, 2012 Amazon ELB Service Event in the US-East Region

We would like to share more details with our customers about the event that occurred with the Amazon Elastic Load Balancing Service ("ELB") earlier this week in the US-East Region. While the service disruption only affected applications using the ELB service (and only a fraction of the ELB load balancers were affected), the impacted load balancers saw significant impact for a prolonged period of time.

The service disruption began at 12:24 PM PST on December 24th when a portion of the ELB state data was logically deleted. This data is used and maintained by the ELB control plane to manage the configuration of the ELB load

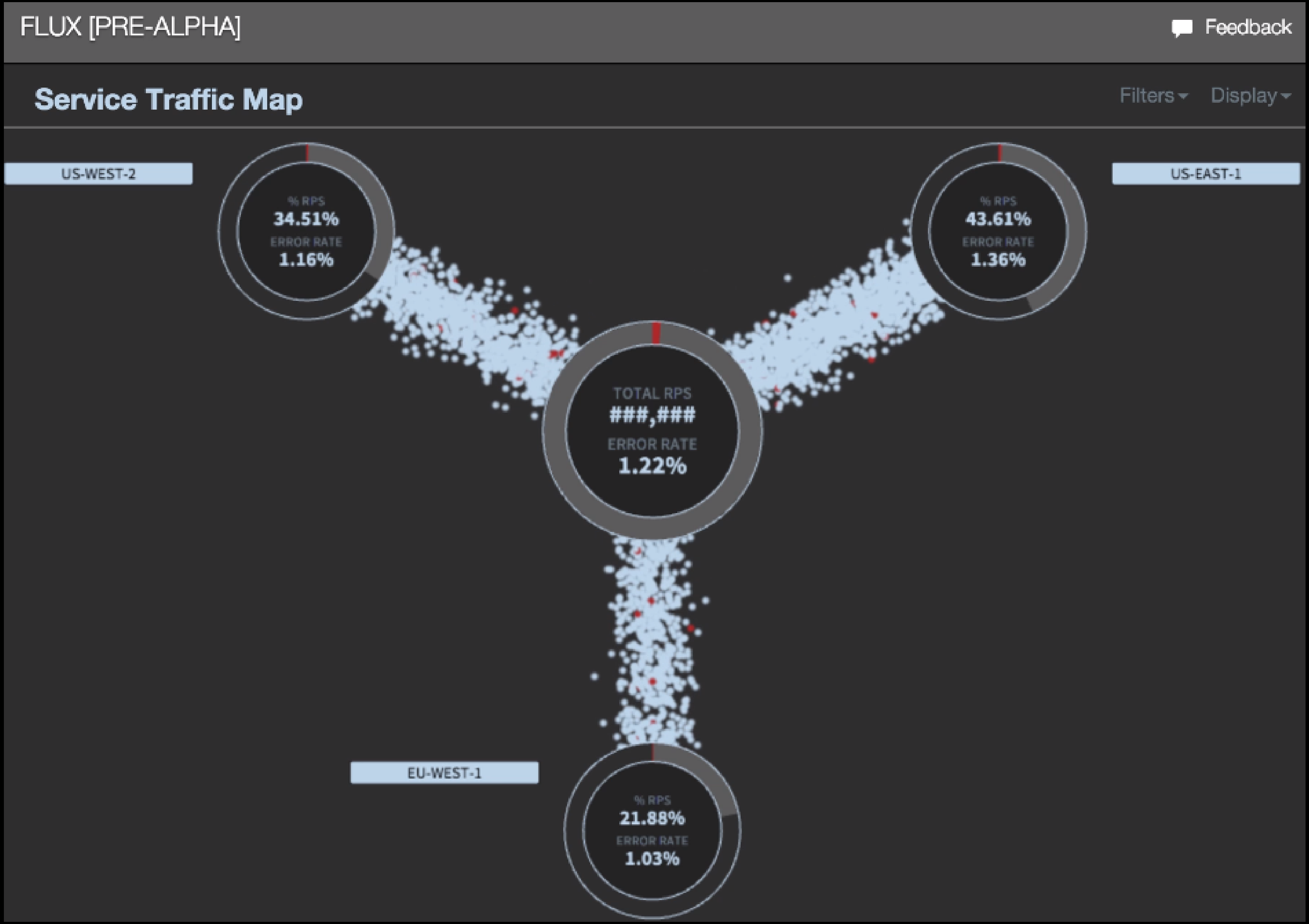
Active-Active

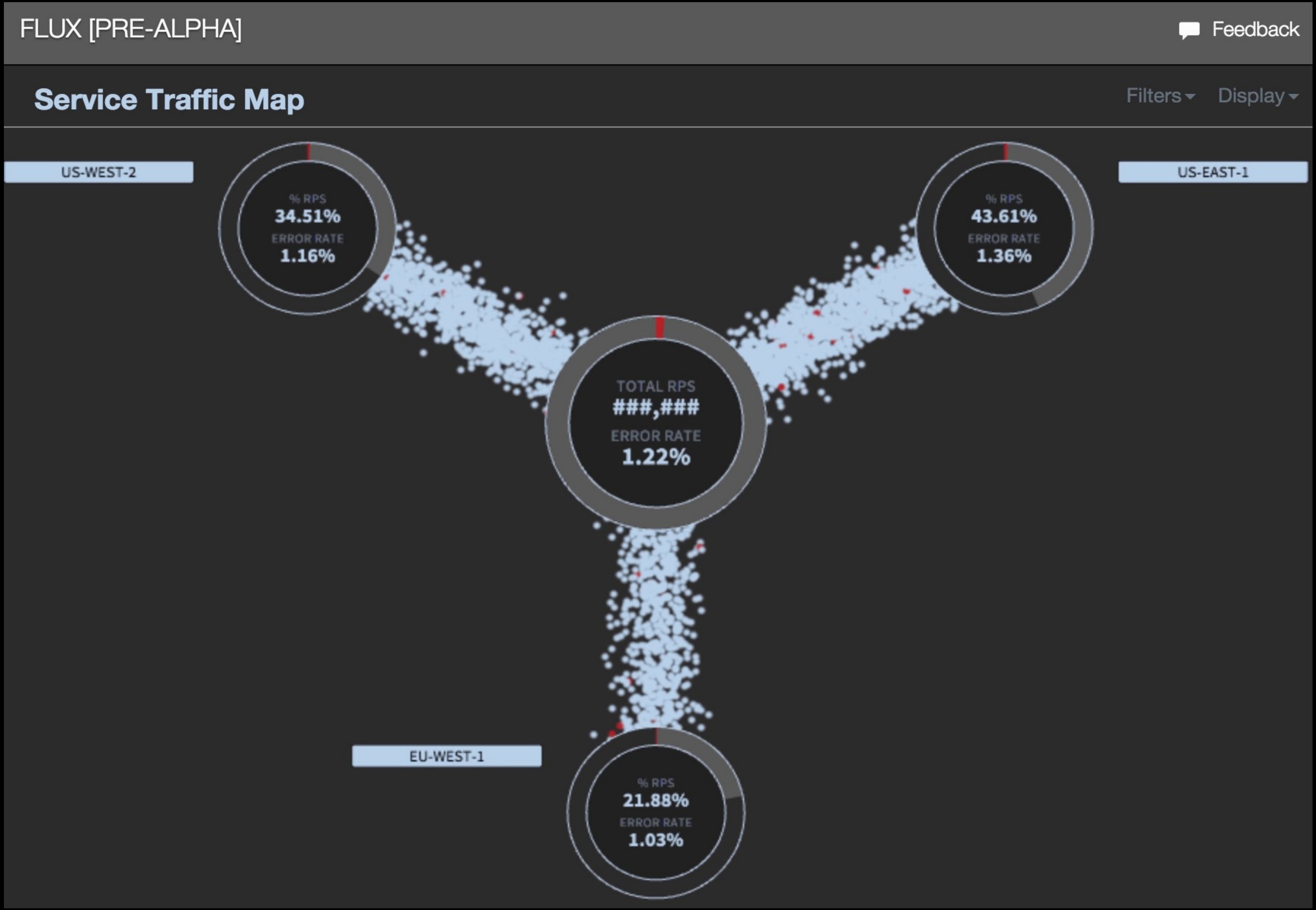
Opportunity

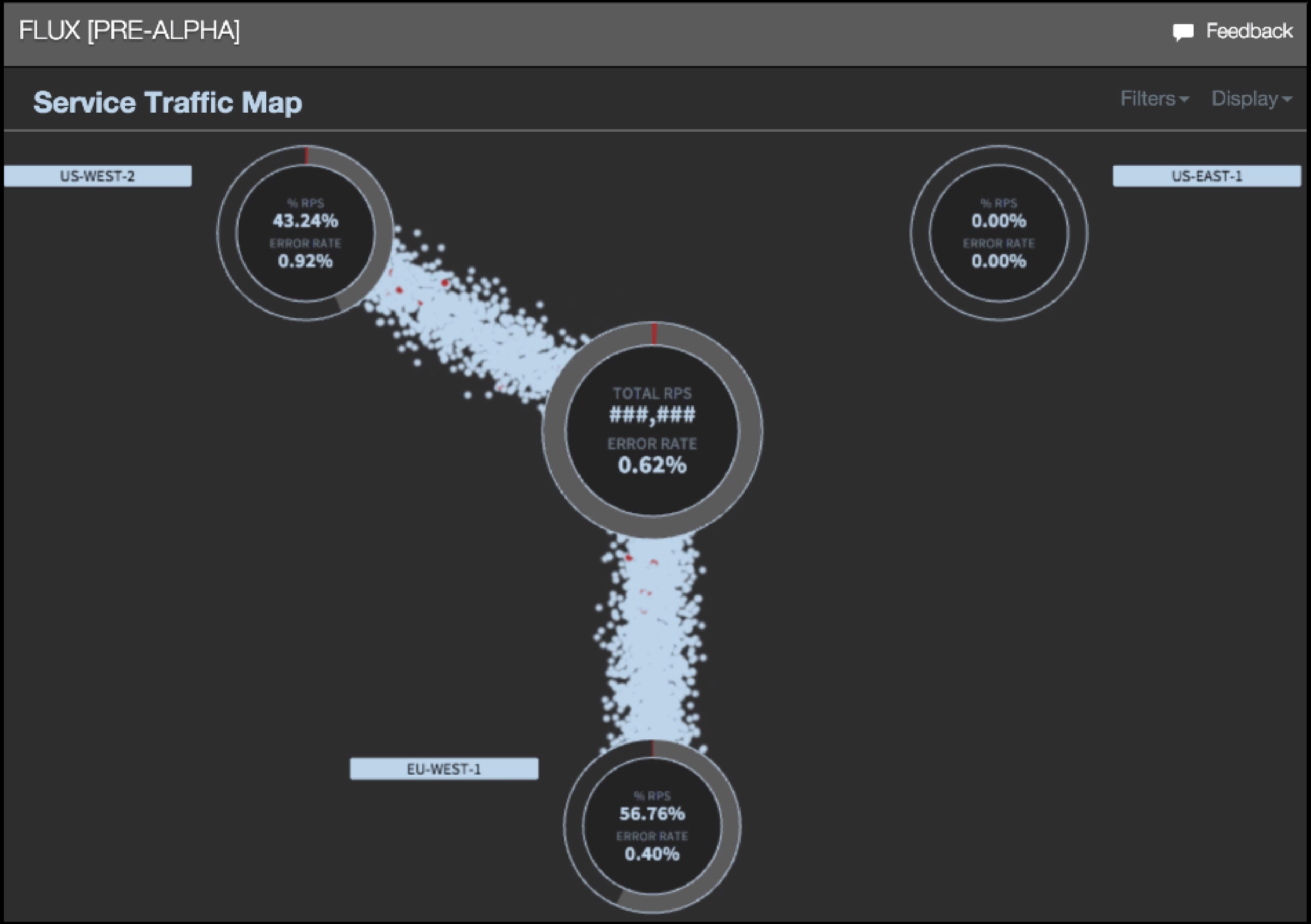
Flow

Fail Out of US-East-1: Case Study

- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation


FLUX [PRE-ALPHA]
Feedback
Service Traffic Map
Filters
Display
US-WEST-2
% RPS
34.51%
ERROR RATE
1.16%
US-EAST-1
% RPS
43.61%
ERROR RATE
1.36%
TOTAL RPS
###,###
ERROR RATE
1.22%
EU-WEST-1
% RPS
21.88%
ERROR RATE
1.03%

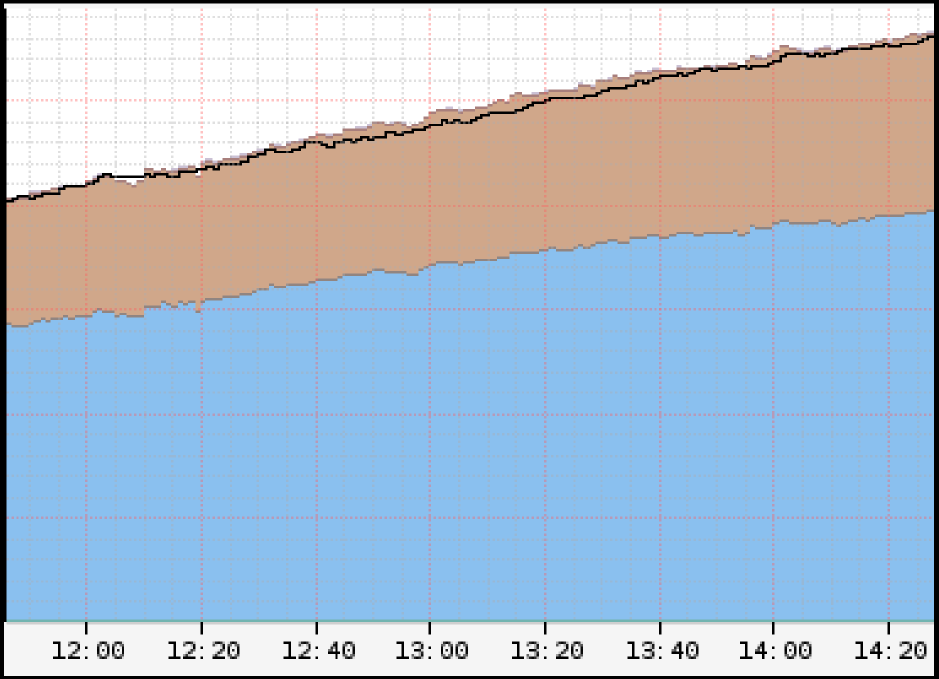
FLUX [PRE-ALPHA]
Feedback
Service Traffic Map
Filters
Display
US-WEST-2
% RPS
34.51%
ERROR RATE
1.16%
US-EAST-1
% RPS
43.61%
ERROR RATE
1.36%
TOTAL RPS
###,###
ERROR RATE
1.22%
EU-WEST-1
% RPS
21.88%
ERROR RATE
1.03%


FLUX [PRE-ALPHA]
Feedback
Service Traffic Map
Filters
Display
US-WEST-2
% RPS
43.24%
ERROR RATE
0.92%
US-EAST-1
% RPS
0.00%
ERROR RATE
0.00%
TOTAL RPS
###,###
ERROR RATE
0.62%
EU-WEST-1
% RPS
56.76%
ERROR RATE
0.40%

Fail Out of US-East-1: Case Study

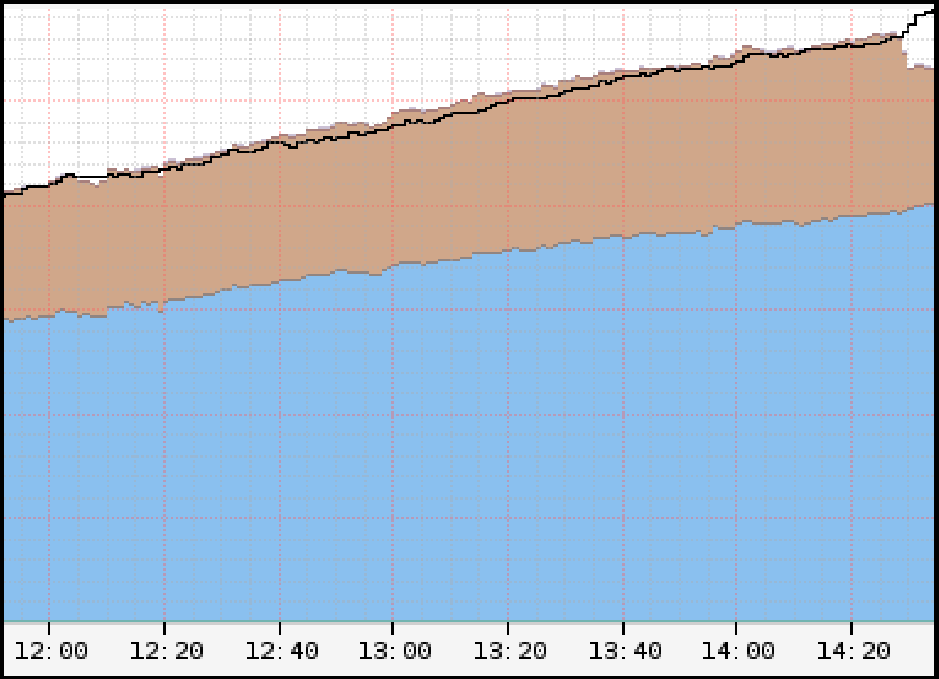
- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation

January 14, 2016



Stream Starts per Second – us-east region

January 14, 2016

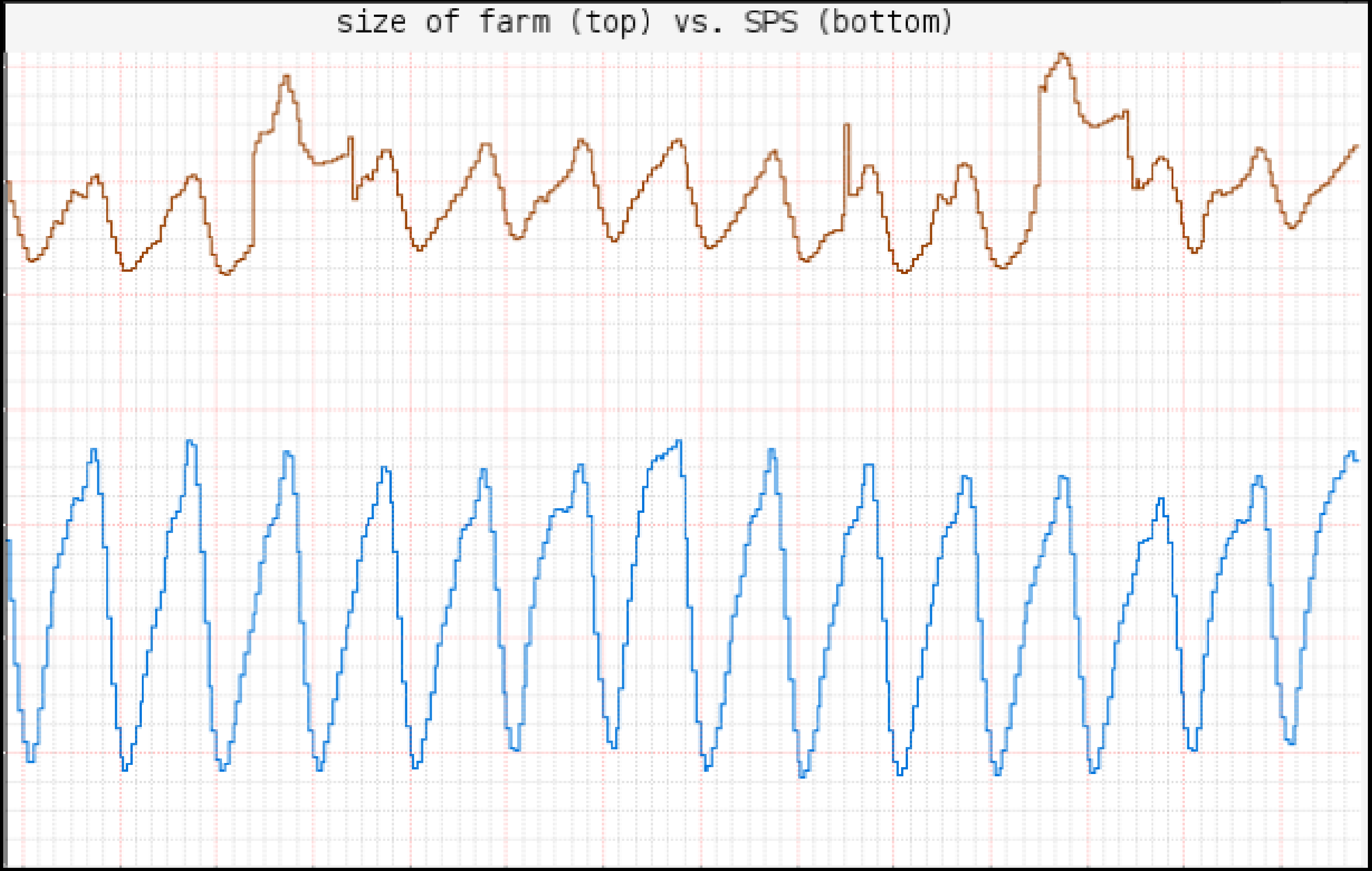


Stream Starts per Second – us-east region

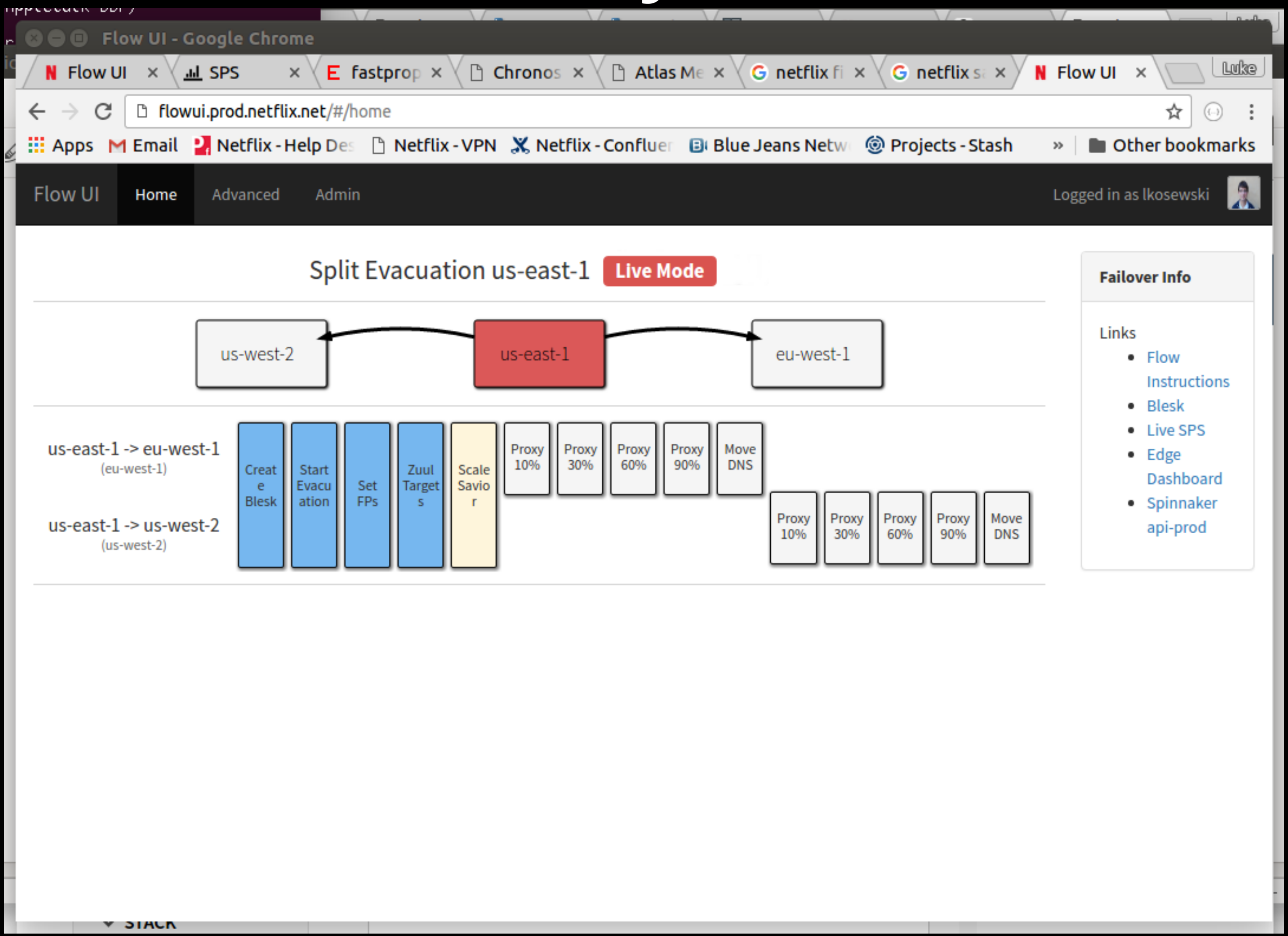
Fail Out of US-East-1: Case Study

- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation

Diurnal Scaling



Y’all Ready for This?



Flow UI - Google Chrome

Split Evacuation us-east-1 **Live Mode**

us-east-1 -> eu-west-1 (eu-west-1)

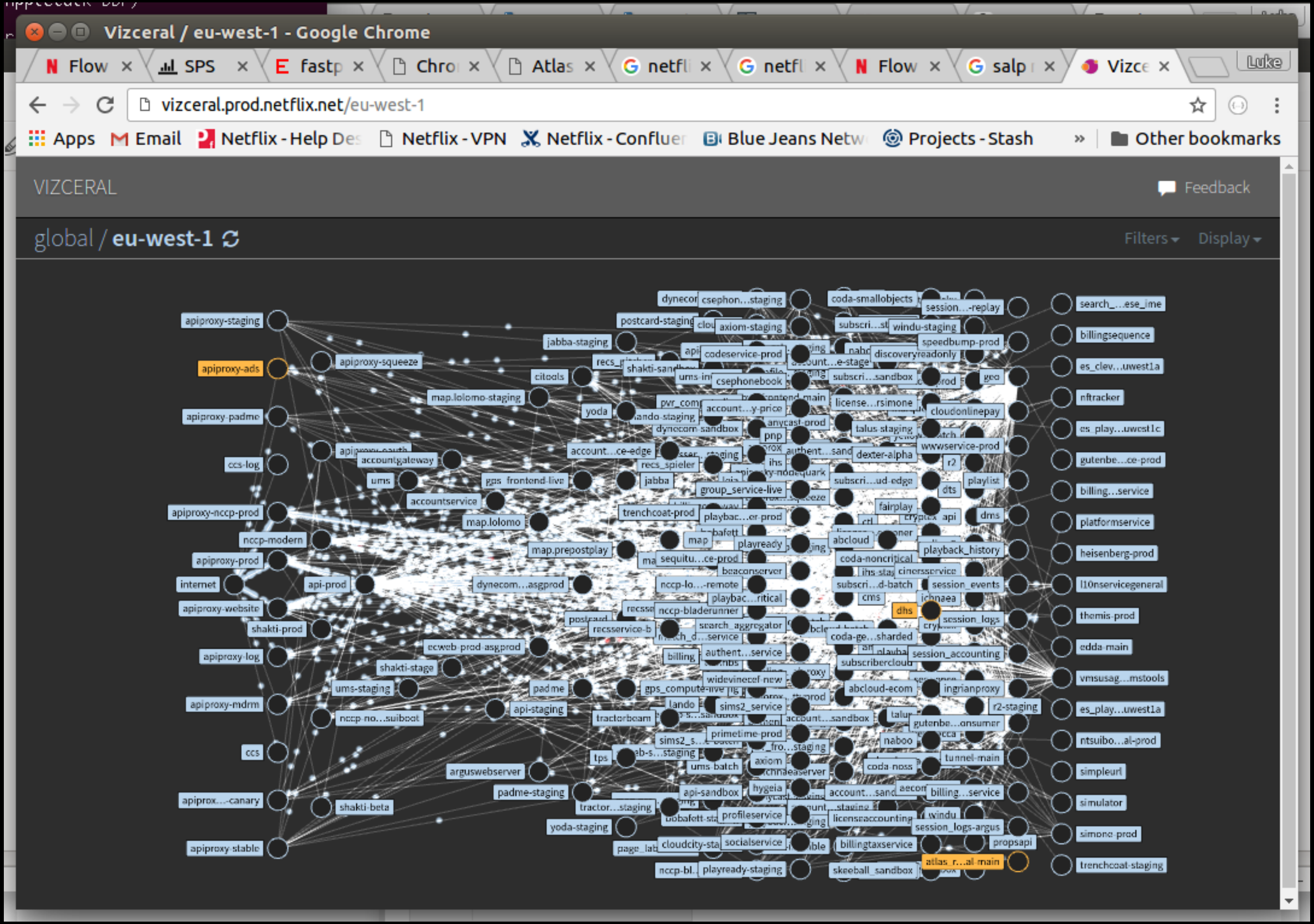
us-east-1 -> us-west-2 (us-west-2)

Failover Info

Links

- Flow Instructions
- Blesk
- Live SPS
- Edge Dashboard
- Spinnaker api-prod

What to Scale?



What to Scale?

- Anything absorbing incoming traffic

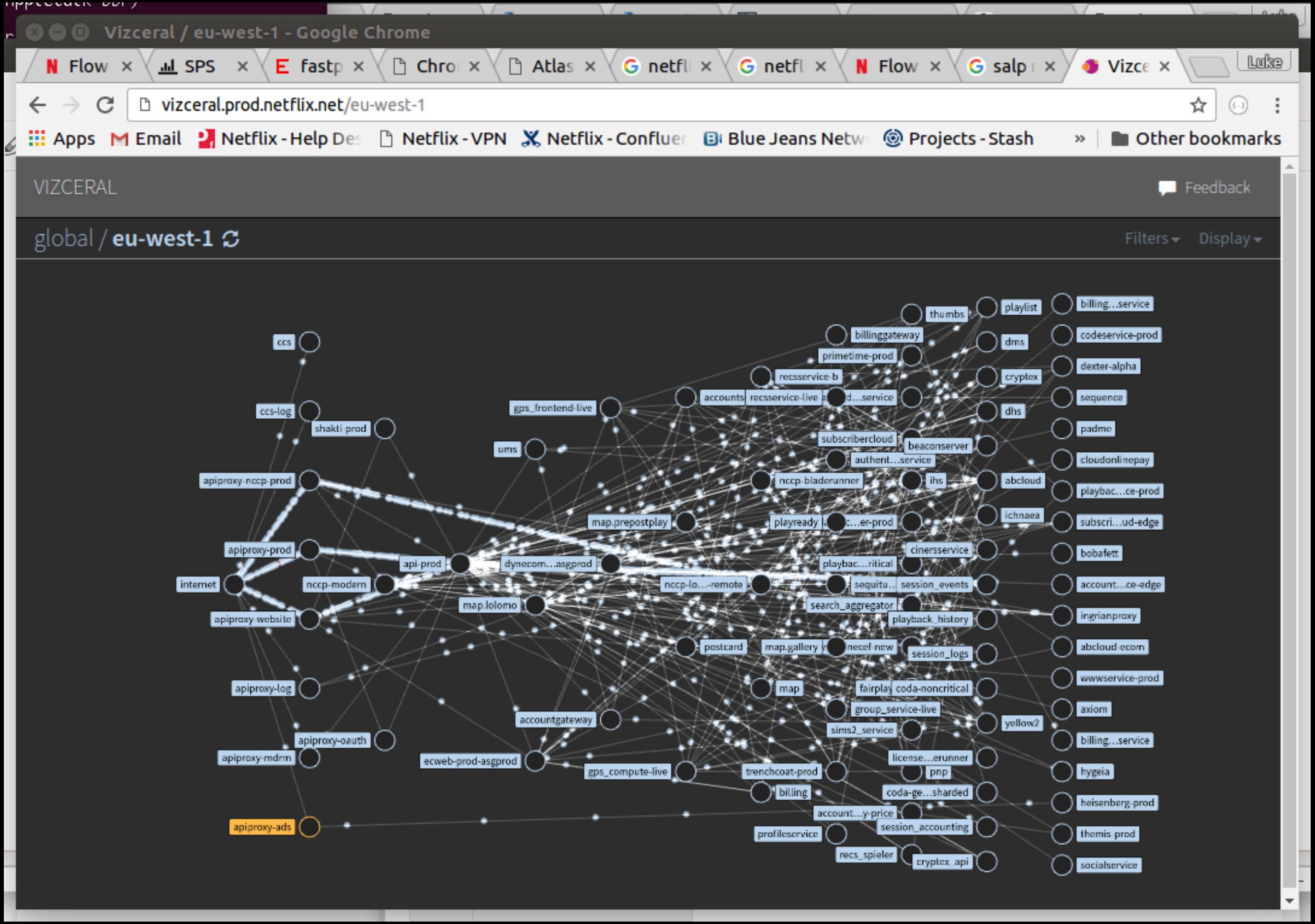
What to Scale?

- Anything absorbing incoming traffic
- Large stateless services

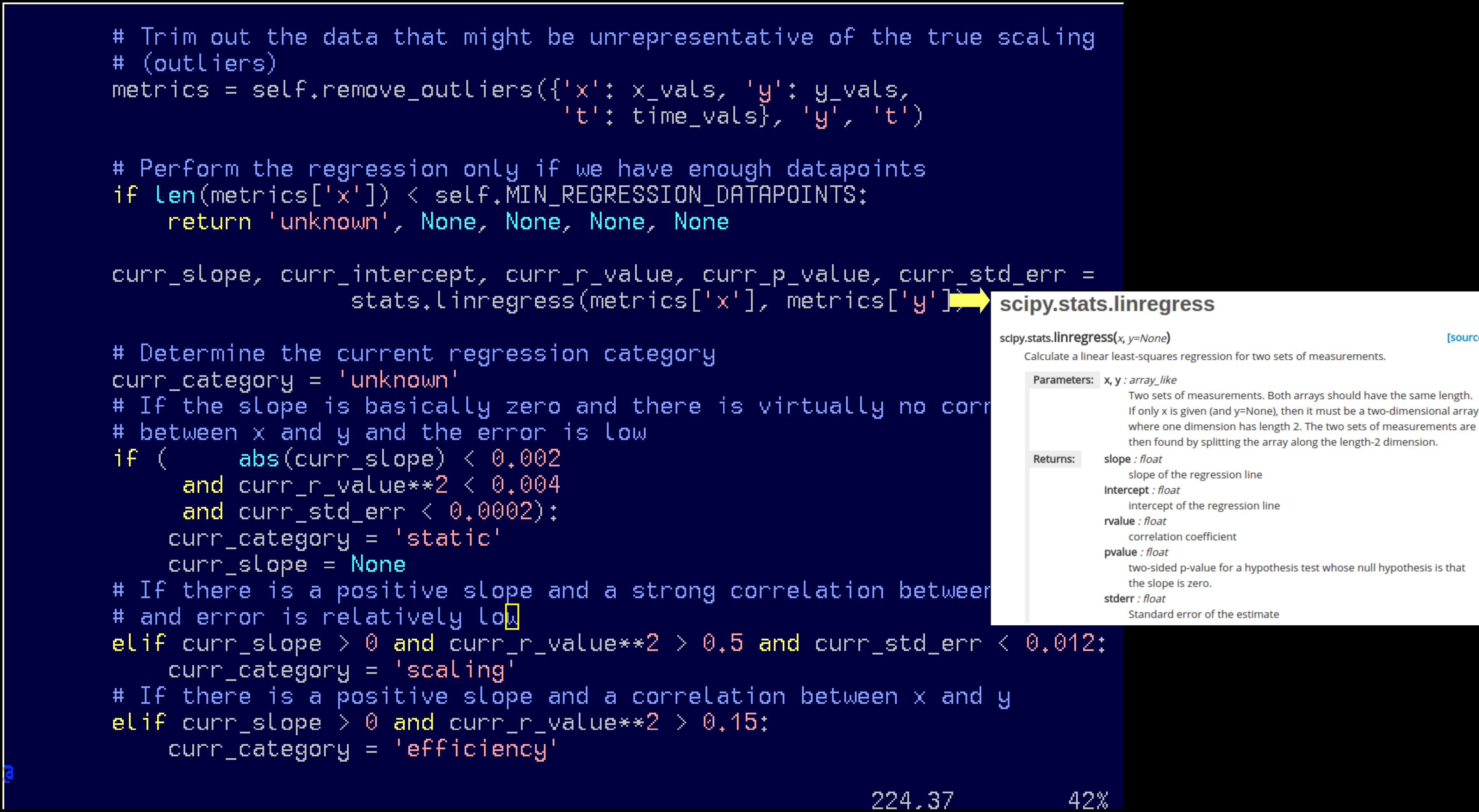
What to Scale?

- Anything absorbing incoming traffic
- Large stateless services
- Required stateful services (carefully)

That’s Better



How to Scale?



```
# Trim out the data that might be unrepresentative of the true scaling
# (outliers)
metrics = self.remove_outliers({'x': x_vals, 'y': y_vals,
                                'y': time_vals}, 'y', 't')

# Perform the regression only if we have enough datapoints
if len(metrics['x']) < self.MIN_REGRESSION_DATAPOINTS:
    return 'unknown', None, None, None, None

curr_slope, curr_intercept, curr_r_value, curr_p_value, curr_std_err =
                 stats.linregress(metrics['x'], metrics['y'])

# Determine the current regression category
curr_category = 'unknown'
# If the slope is basically zero and there is virtually no corr
# between x and y and the error is low
if (     abs(curr_slope) < 0.002
     and curr_r_value**2 < 0.004
     and curr_std_err < 0.0002):
    curr_category = 'static'
    curr_slope = None
# If there is a positive slope and a strong correlation betweer
# and error is relatively low
elif curr_slope > 0 and curr_r_value**2 > 0.5 and curr_std_err < 0.012:
    curr_category = 'scaling'
# If there is a positive slope and a correlation between x and y
elif curr_slope > 0 and curr_r_value**2 > 0.15:
    curr_category = 'efficiency'

                                                              224,37         42%
```

scipy.stats.linregress

scipy.stats.**linregress**(*x*, *y=None*) [sourc

Calculate a linear least-squares regression for two sets of measurements.

Parameters: **x, y** : *array_like*

Two sets of measurements. Both arrays should have the same length. If only x is given (and y=None), then it must be a two-dimensional array where one dimension has length 2. The two sets of measurements are then found by splitting the array along the length-2 dimension.

Returns: **slope** : *float*

slope of the regression line

intercept : *float*

intercept of the regression line

rvalue : *float*

correlation coefficient

pvalue : *float*

two-sided p-value for a hypothesis test whose null hypothesis is that the slope is zero.

stderr : *float*

Standard error of the estimate

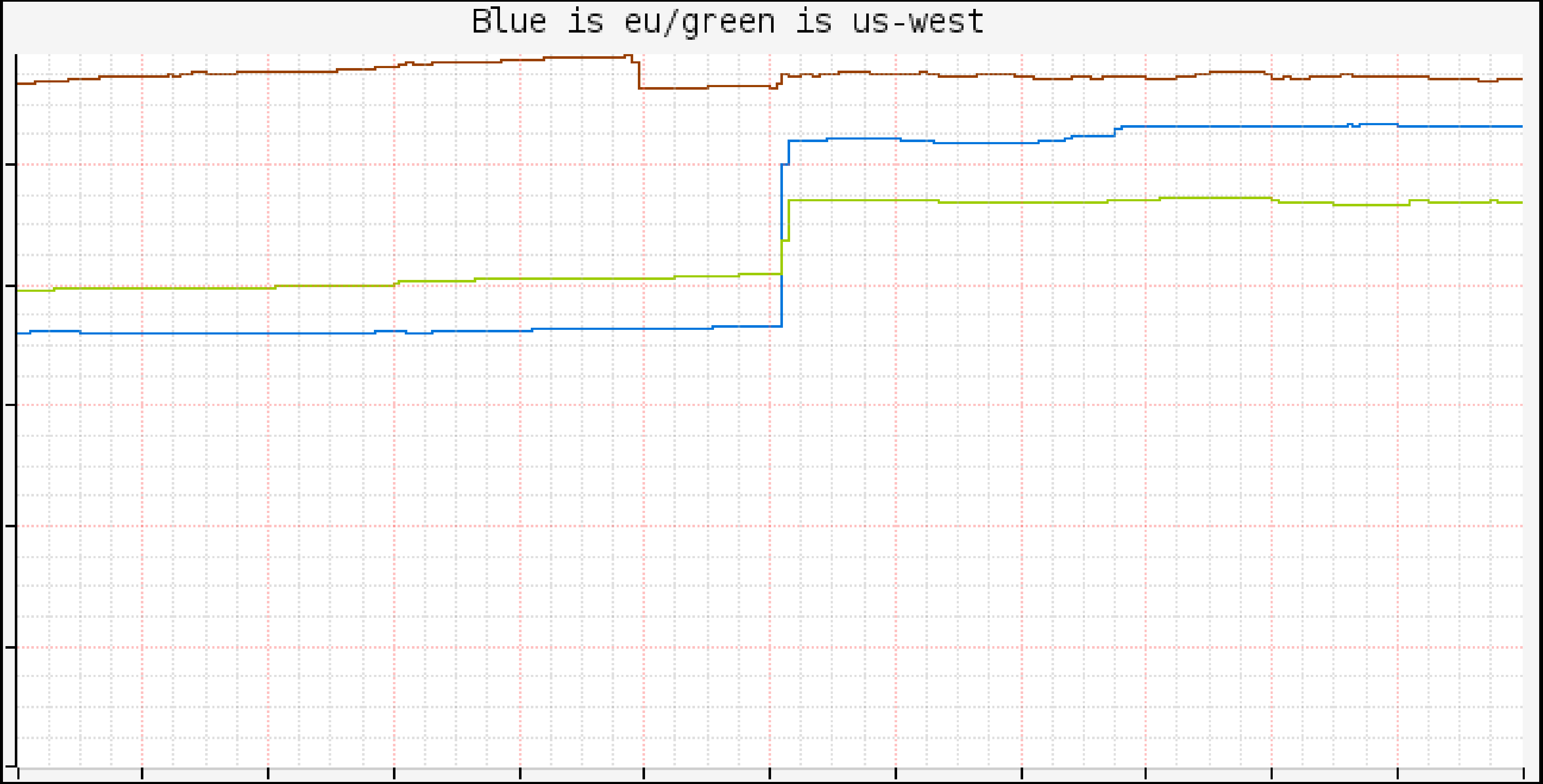
Two More Fallbacks

- “Time of Day” estimation

Two More Fallbacks

- “Time of Day” estimation
- largest observed value in the last 24h as an intercept

How Much?



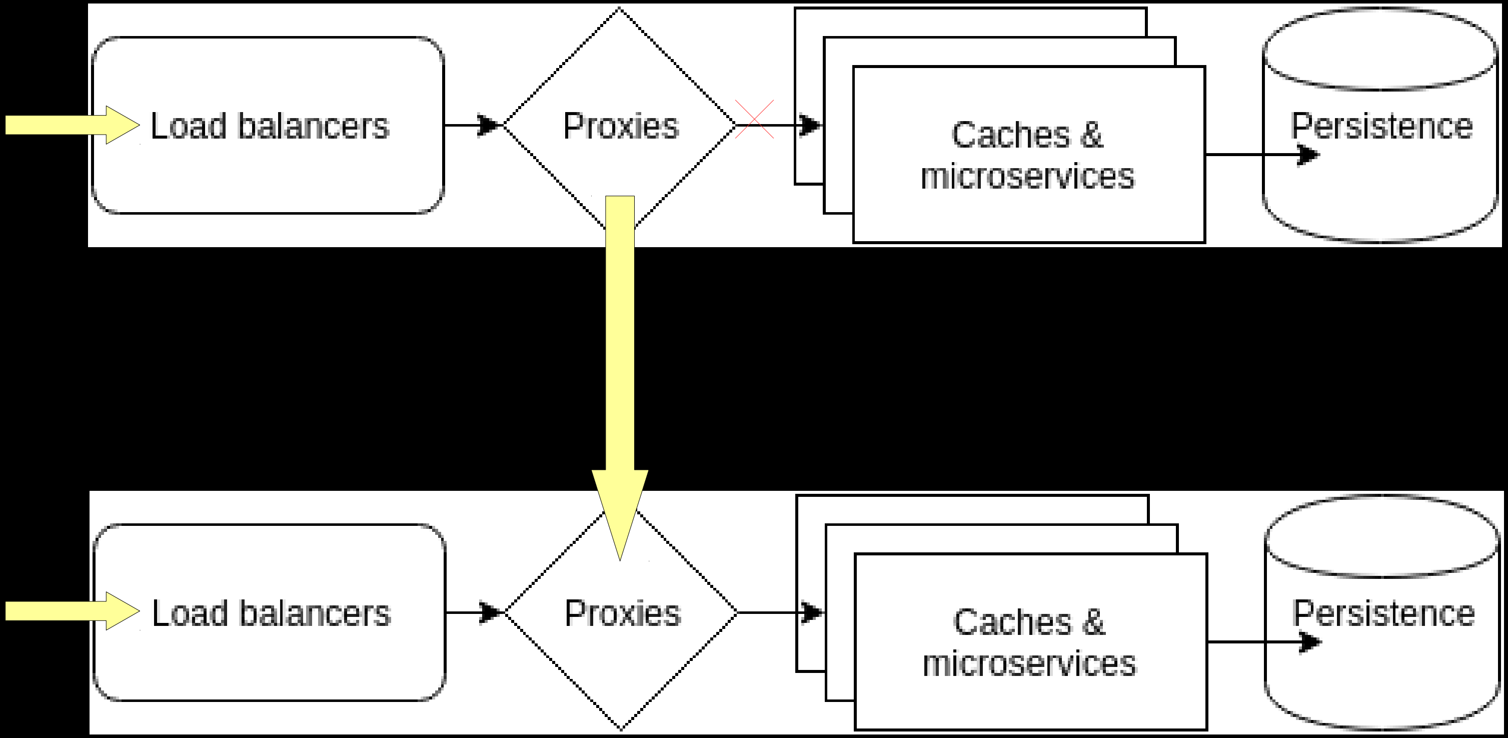
Ooze

Nimble

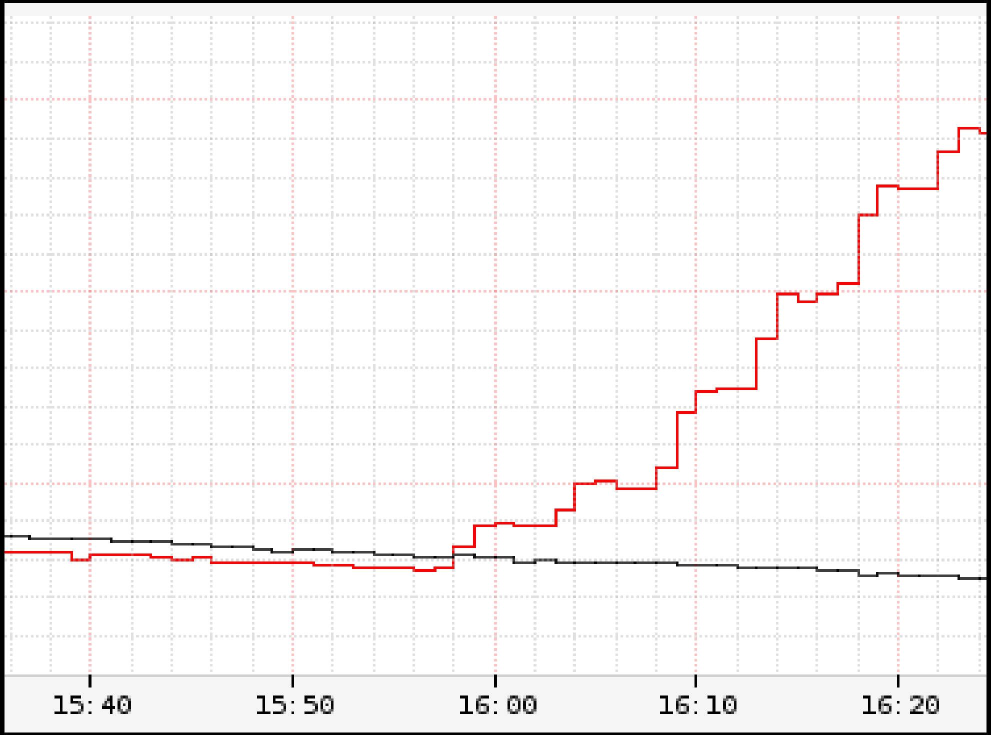
Fail Out of US-East-1: Case Study

- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation

What do I mean by that?



Why We Proxy



Stream Starts per Second - EU

How do We Proxy?

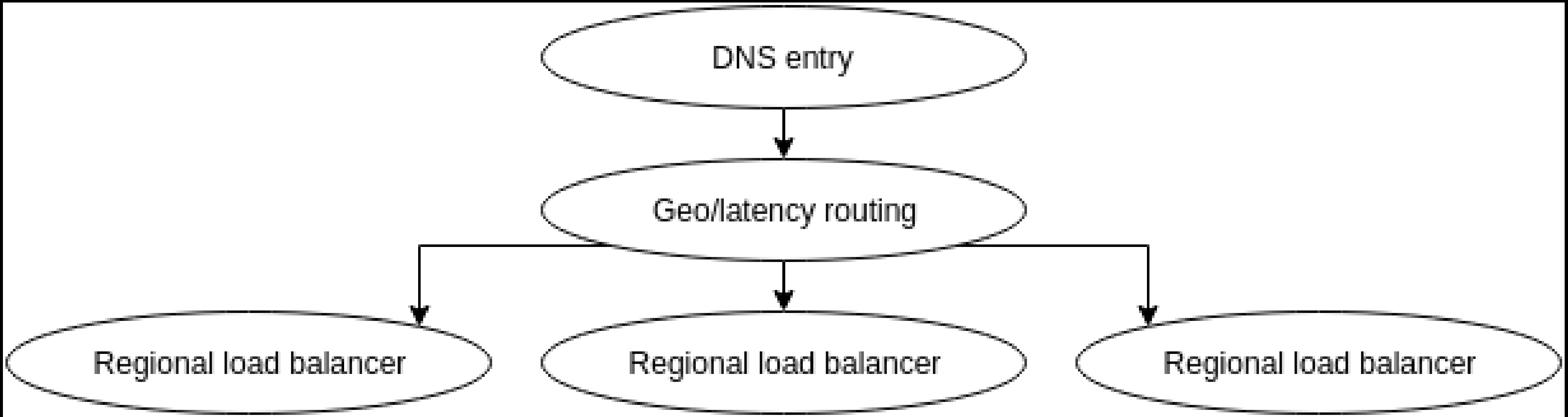
Archaius dynamic properties – regionally scoped

Zuul proxy with dynamic filters (Groovy)

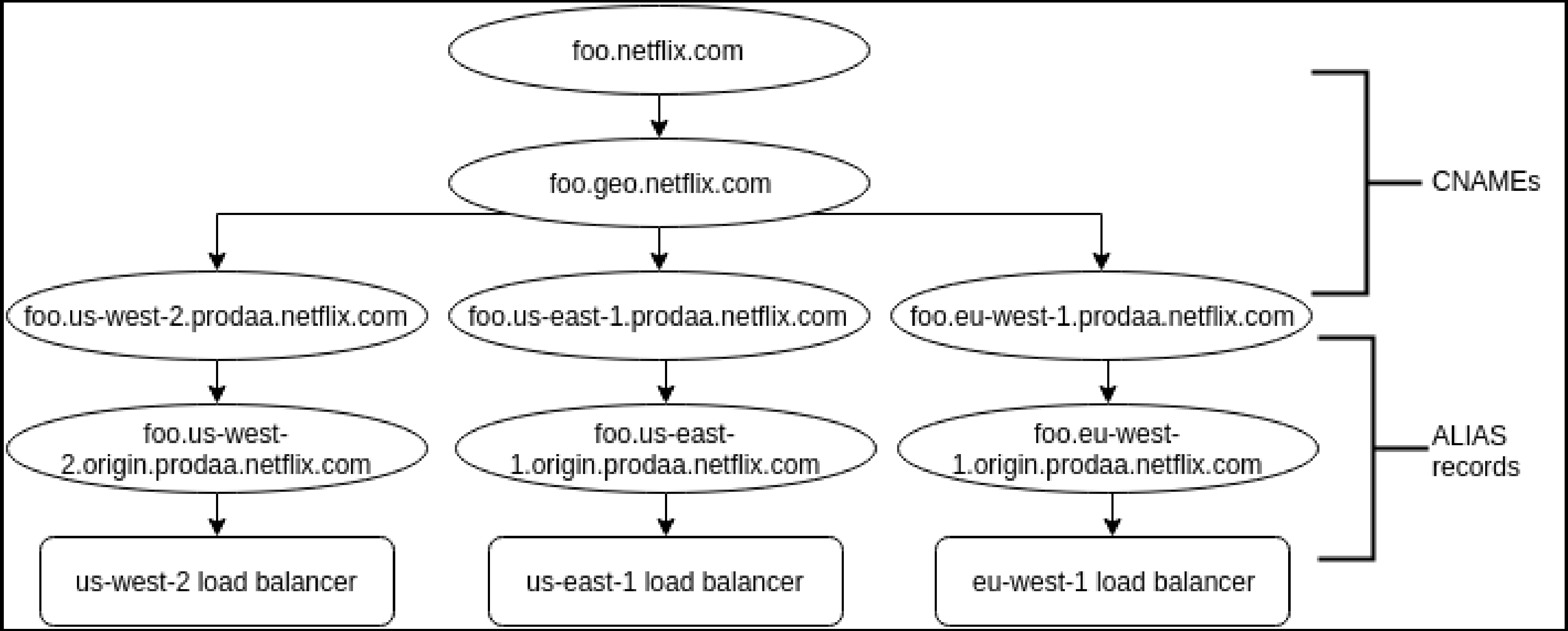
Fail Out of US-East-1: Case Study

- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation

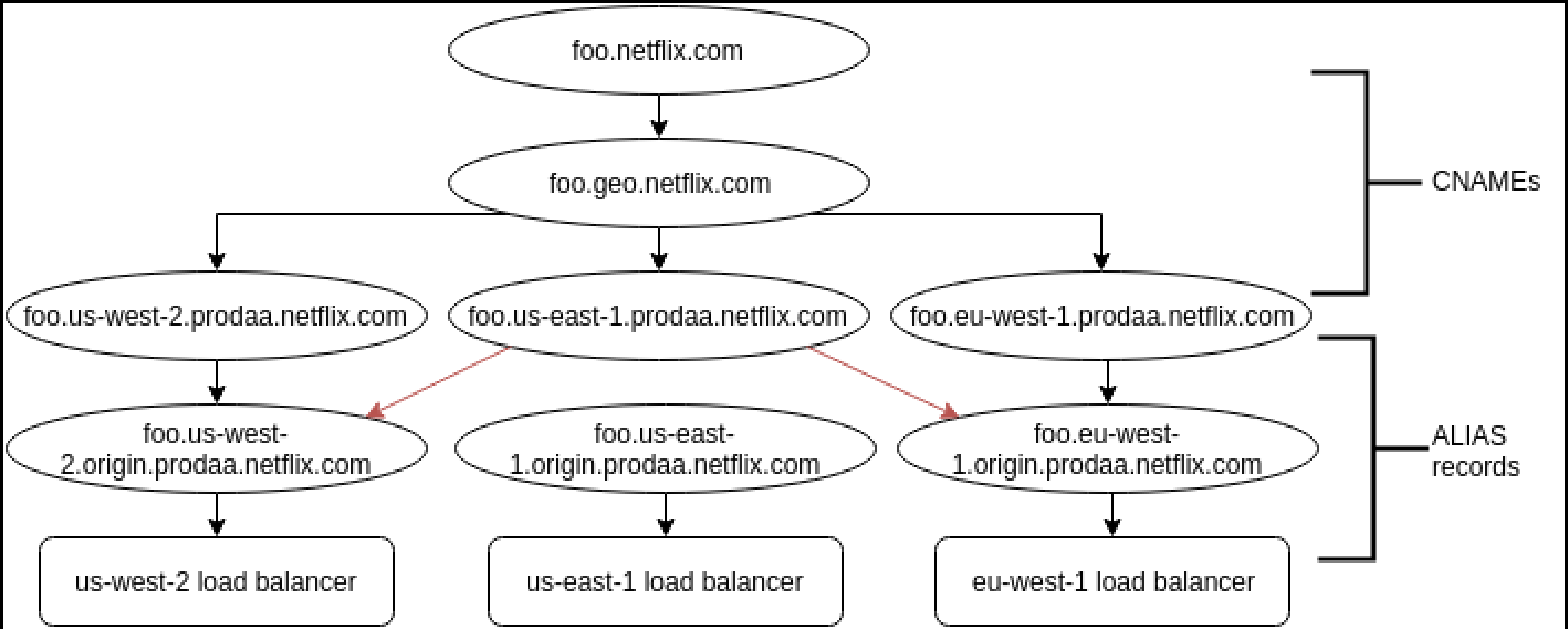
Traditional DNS



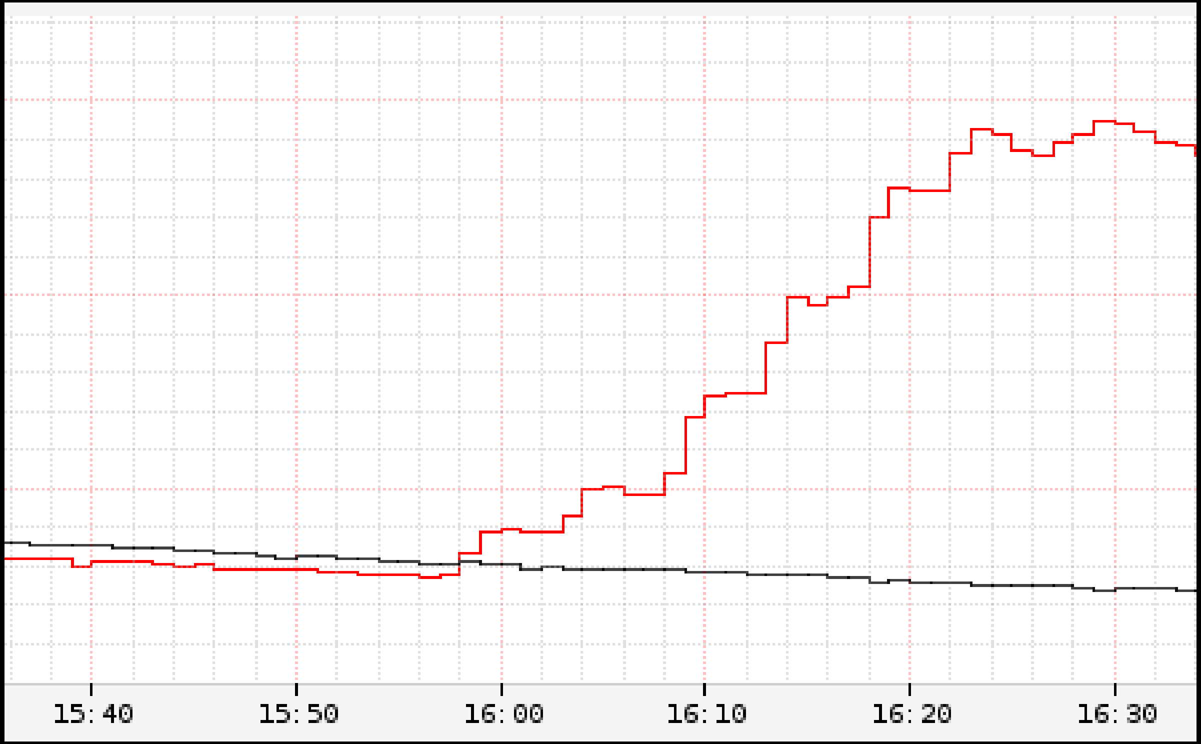
Netflix’s DNS as a DB


foo.netflix.com
foo.geo.netflix.com
foo.us-west-2.prodaa.netflix.com
foo.us-east-1.prodaa.netflix.com
foo.eu-west-1.prodaa.netflix.com
foo.us-west-2.origin.prodaa.netflix.com
foo.us-east-1.origin.prodaa.netflix.com
foo.eu-west-1.origin.prodaa.netflix.com
us-west-2 load balancer
us-east-1 load balancer
eu-west-1 load balancer
CNAMEs
ALIAS records

Failover

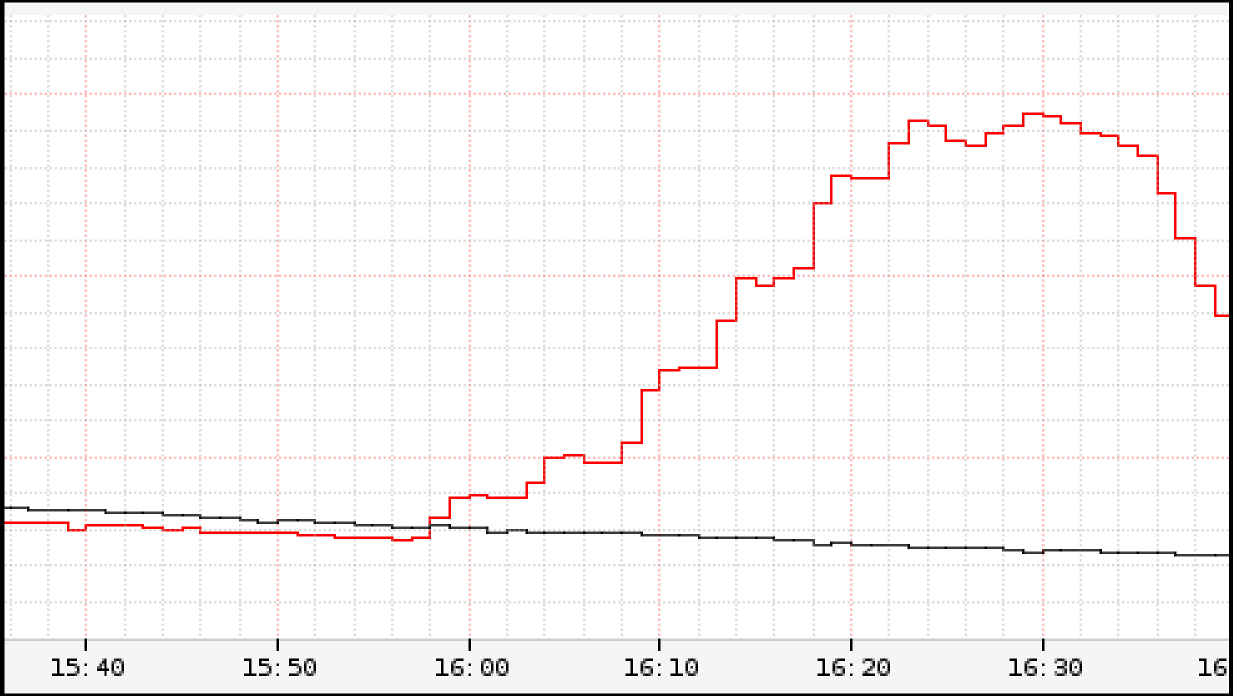

foo.netflix.com
foo.geo.netflix.com
foo.us-west-2.prodaa.netflix.com
foo.us-east-1.prodaa.netflix.com
foo.eu-west-1.prodaa.netflix.com
foo.us-west-2.origin.prodaa.netflix.com
foo.us-east-1.origin.prodaa.netflix.com
foo.eu-west-1.origin.prodaa.netflix.com
us-west-2 load balancer
us-east-1 load balancer
eu-west-1 load balancer
CNAMEs
ALIAS records

Are We Done?



Stream Starts per Second - EU

Nope

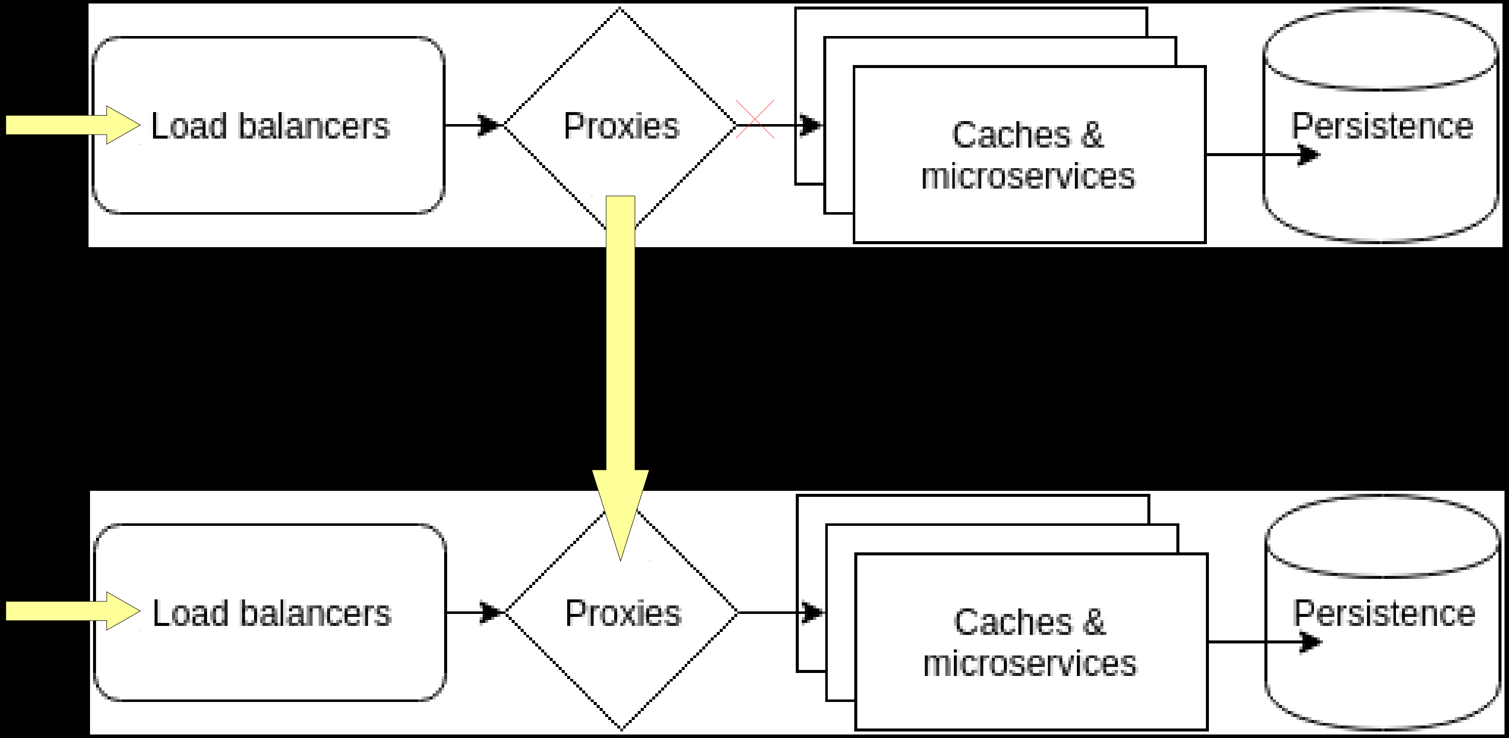


Stream Starts per Second - EU

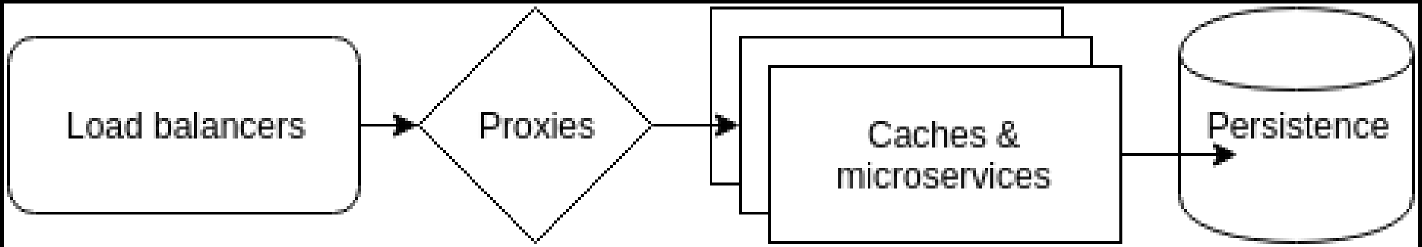
Fail Out of US-East-1: Case Study

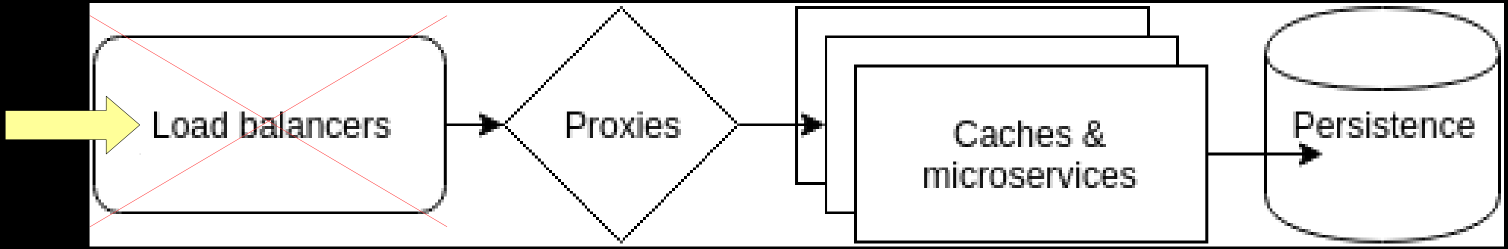
- Outage!
- Scaling-up
- Proxying
- DNS design and cutover
- Improvisation

Recap: Proxying

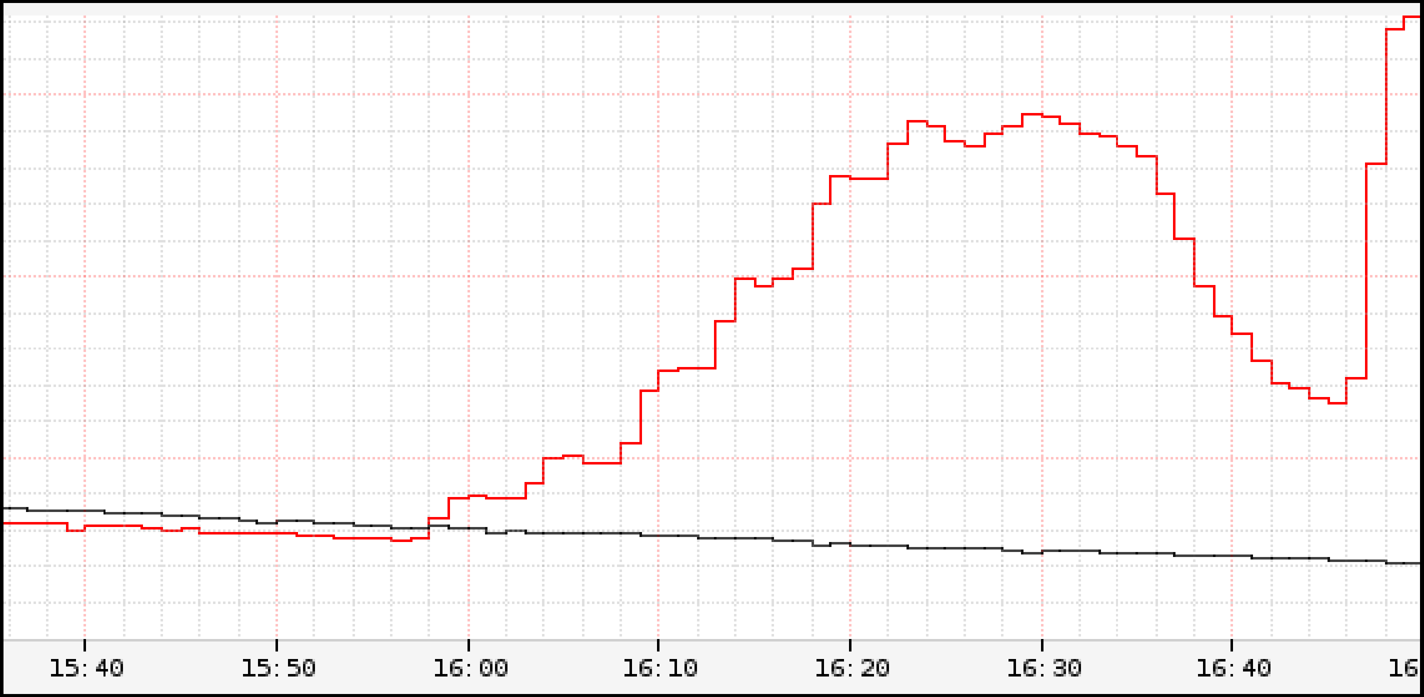


Recap: Proxying



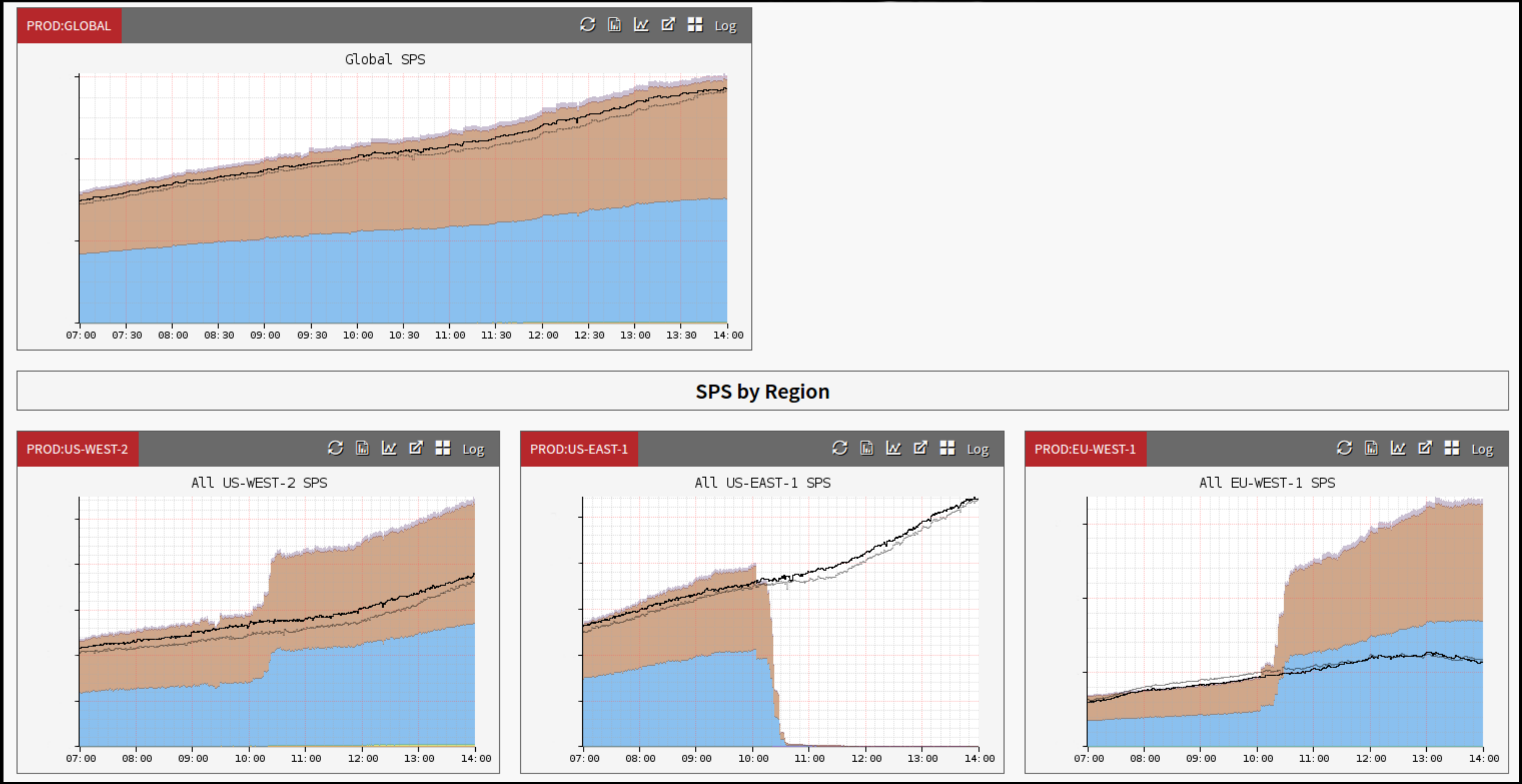


The Crowd Goes Wild



Stream Starts per Second - EU

This is What Success Feels Like



Positive Feedback Loop

The more we practice, the better and more daring we get

Other Takeaways

NETFLIX

Thank You and Questions

Luke Kosewski – luke@netflix.com
Traffic & Chaos Engineering

Summary of NFLX github/techblog links

- Active/Active
  http://techblog.netflix.com/2013/12/active-active-for-multi-regional.html
  http://techblog.netflix.com/2016/03/global-cloud-active-active-and-beyond.html
- Archaius
  https://github.com/Netflix/archaius
  http://techblog.netflix.com/2012/06/annoucing-archaius-dynamic-properties.html
- Zuul
  https://github.com/Netflix/zuul
  http://techblog.netflix.com/2013/06/announcing-zuul-edge-service-in-cloud.html
- SPS
  http://techblog.netflix.com/2015/02/sps-pulse-of-netflix-streaming.html