

Chaos Engineering at Netflix

Lorin Hochstein (@lhochstein)

What I've learned doing
chaos at Netflix

Who am I?

3 years on Resilience
Engineering (Chaos) team



Chaos Engineering

Building Confidence in System Behavior
through Experiments



Casey Rosenthal, Lorin Hochstein,
Aaron Blohowiak, Nora Jones
& Ali Basiri

Chaos Engineering

Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production.

-- principlesofchaos.org

Find weaknesses...

...using failure injection...

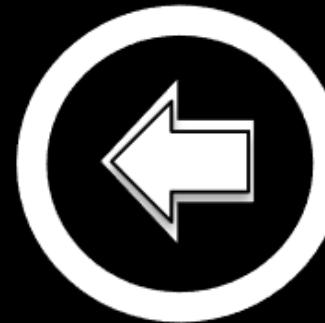
...in production.

A play in four acts

- Act I: Some context about Netflix operations
- Act II: Chaos at Netflix when I got there
- Act III: Chaos as experimentation
- Act IV: Lessons learned

Act I: Some context about Netflix operations

We care about availability



Whoops, something went wrong...

Netflix Streaming Error

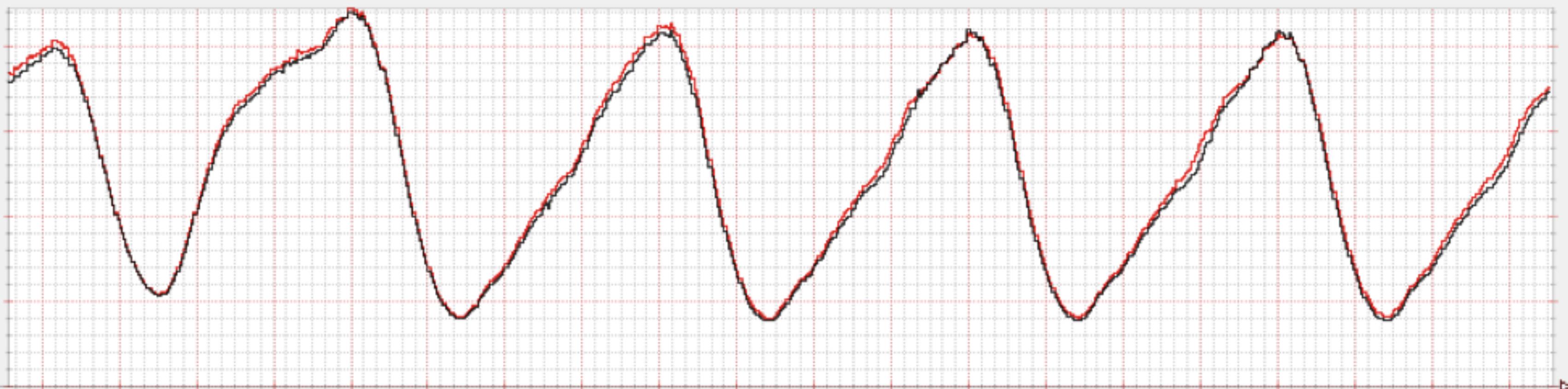
We're having trouble playing this title right now. Please try again later or select a different title.

SPS: Stream starts Per Second

Number of people who hit the “play” button and successfully started

99,95%

SPS



Microservice architecture

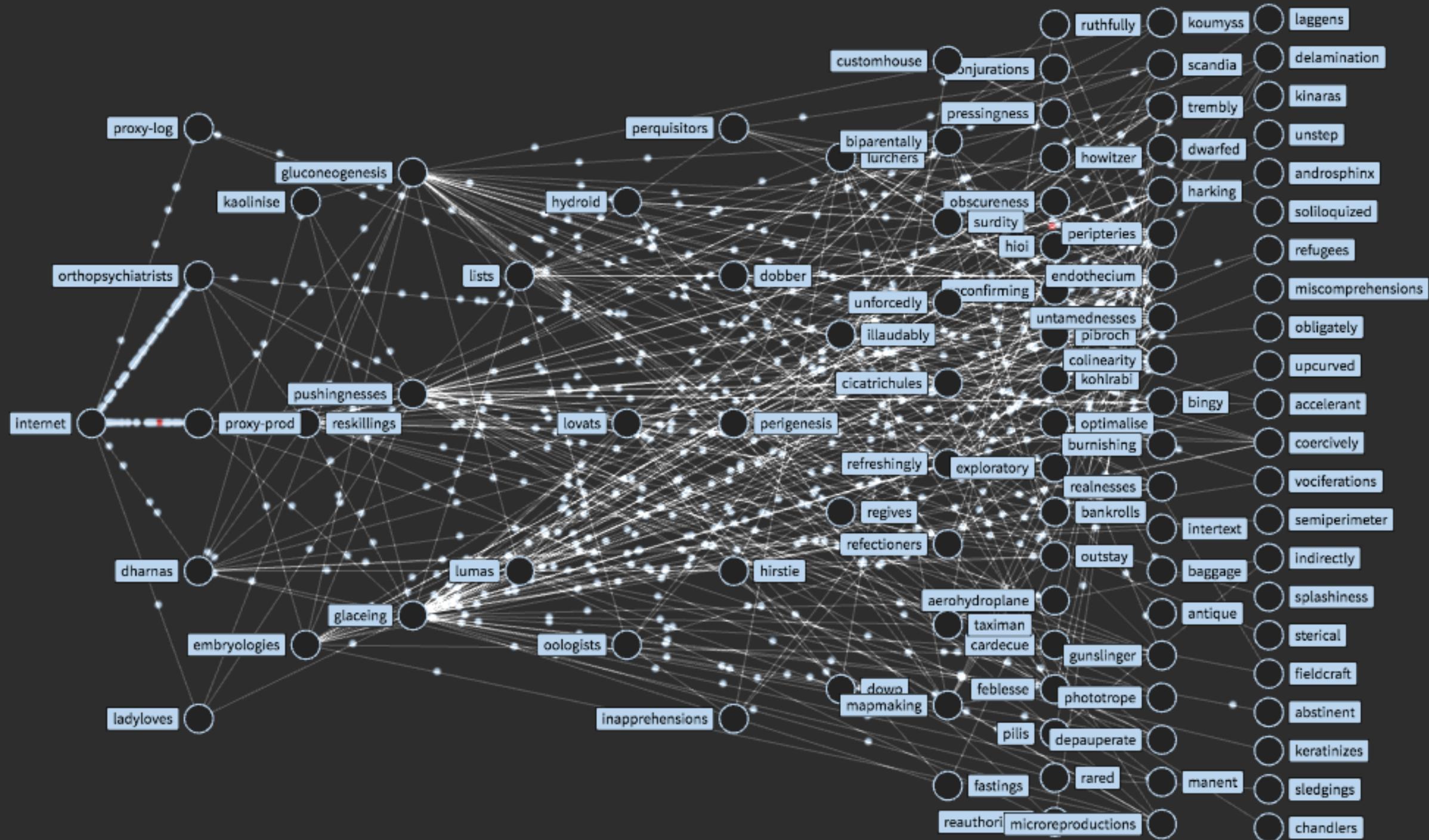
global / us-east-1

99 services / 2 filtered (show)

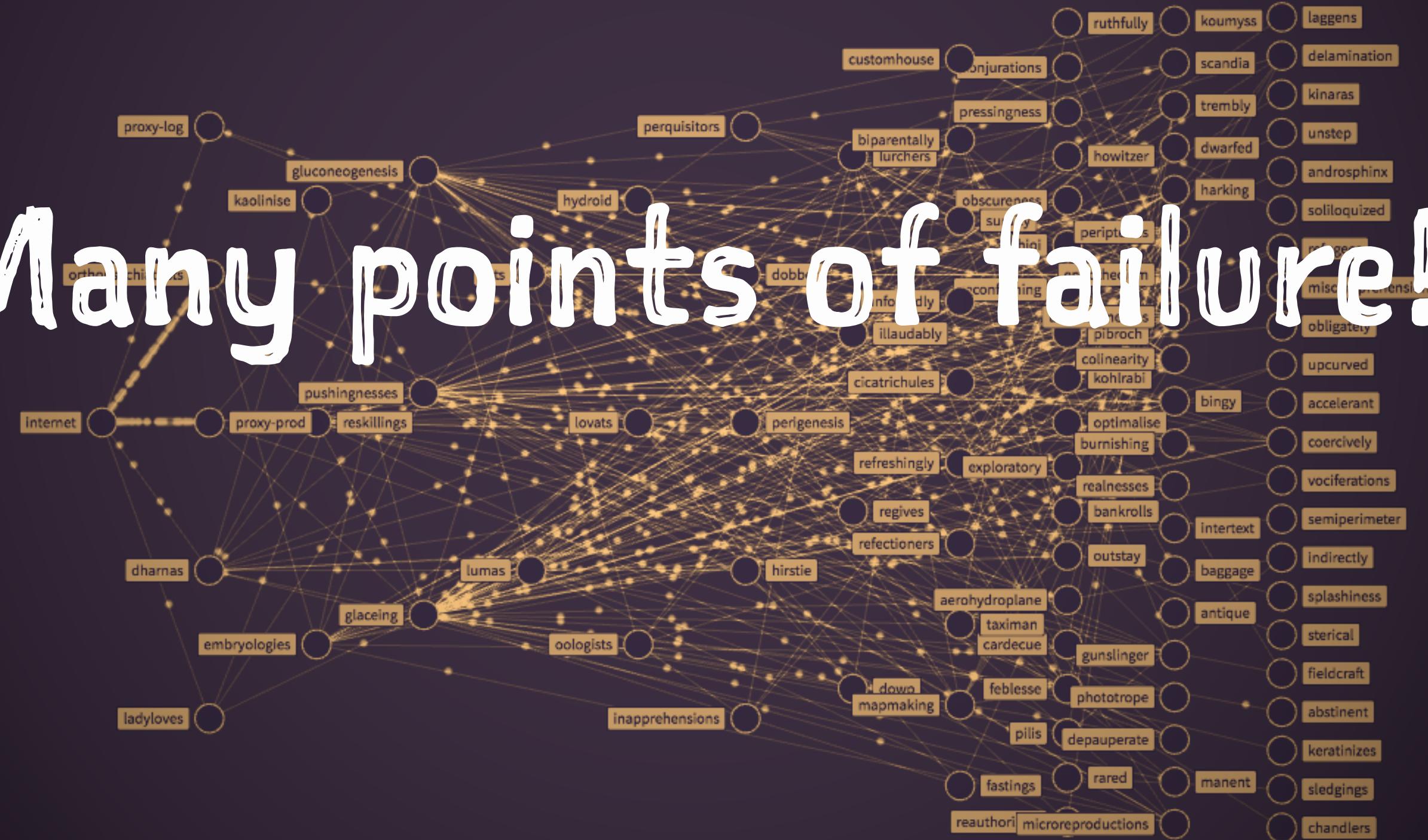
Locate Service



Filters ▾ Display ▾



Many points of failure!



We don't have SLOs

We don't have SLOs 😱

Service-level objectives

95% of requests will be
serviced within 100ms

What failure looks like at Netflix

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

503 Service Unavailable

Resource exhaustion

CPU

Threads

Memory

I/O

500 Internal Server Error?

```
18:18:00,094 ERROR FitContextImpl:195 - Fit Error checking or injecting failure
java.lang.NullPointerException
    at com.netflix.fit.InjectionPointImpl.wildcardMatch(InjectionPointImpl.java:133)
    at com.netflix.fit.scenario.FitScenarioImpl.shouldImpact(FitScenarioImpl.java:45)
    at com.netflix.fit.FitContextImpl.shouldInjectFailure(FitContextImpl.java:130)
    at com.netflix.fit.FitContextImpl.checkAndInjectFailure(FitContextImpl.java:191)
    at com.netflix.fit.FitContext.checkAndInjectFailure(FitContext.java:40)
    at com.netflix.server.base.fit.FitHandler.handle(FitHandler.java:34)
    at com.netflix.server.base.NFFilter.safeDoFilter(NFFilter.java:574)
    at com.netflix.server.base.NFFilter.access$200(NFFilter.java:234)
    at com.netflix.server.base.NFFilter$3.call(NFFilter.java:482)
    at com.netflix.server.base.NFFilter$3.call(NFFilter.java:479)
    at com.netflix.lang.BindingContexts.callWithNewContext(BindingContexts.java:182)
    at com.netflix.server.base.NFFilter.doFilter(NFFilter.java:479)
    at com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:82)
    at com.google.inject.servlet.ManagedFilterPipeline.dispatch(ManagedFilterPipeline.java:120)
    at com.google.inject.servlet.GuiceFilter.doFilter(GuiceFilter.java:135)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:106)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:502)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)
```

Many problems caught by
canaries

Operational change can
leave clusters underscaled

System is resilient to
individual failures

Retries

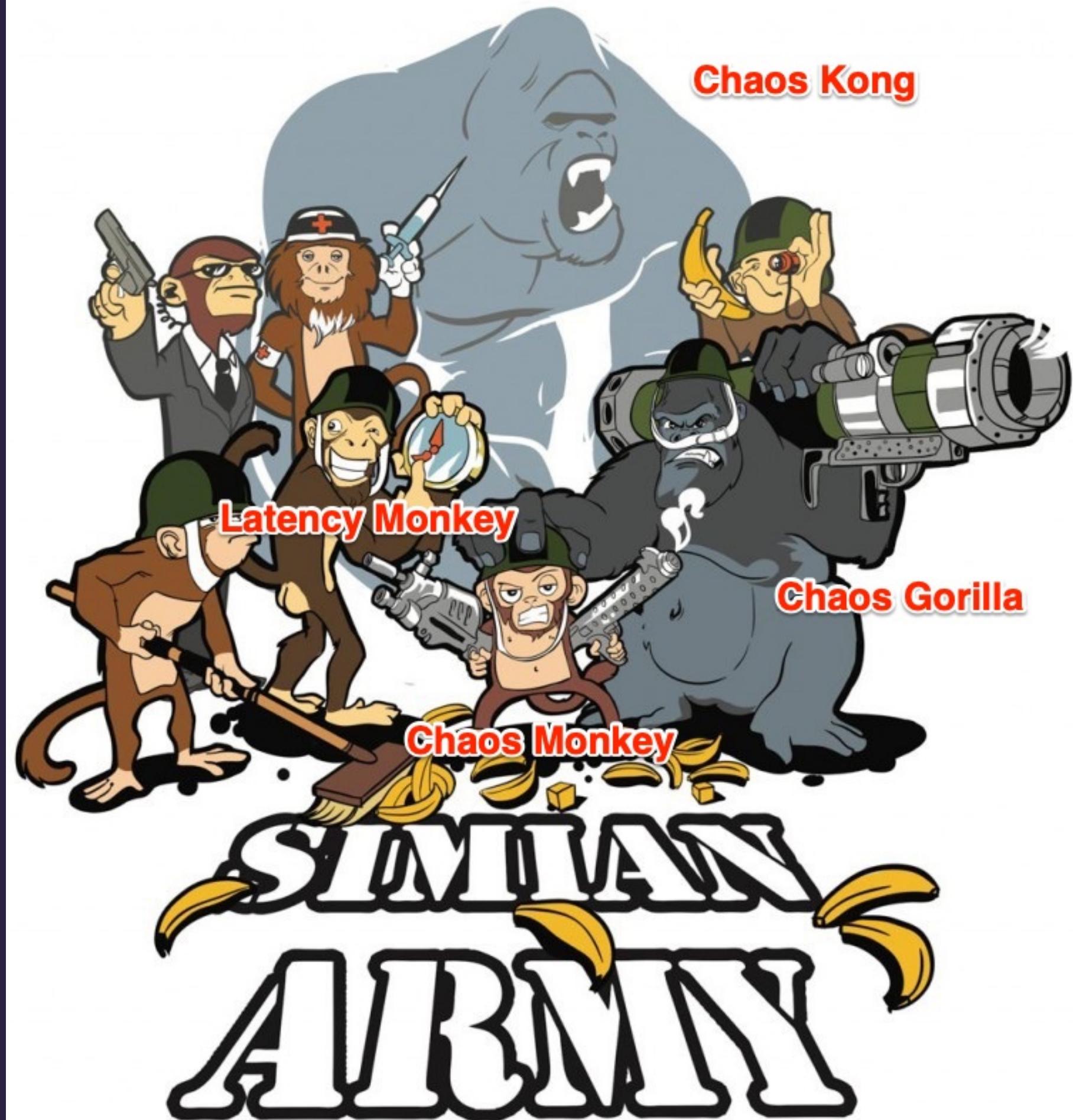
A photograph of a night sky filled with numerous bright, branching lightning bolts. The lightning is concentrated in several horizontal bands across the dark, cloudy sky. In the foreground, the silhouettes of trees are visible against the bright light from the storm. A single street lamp is visible on the right side, casting a small glow.

Retries

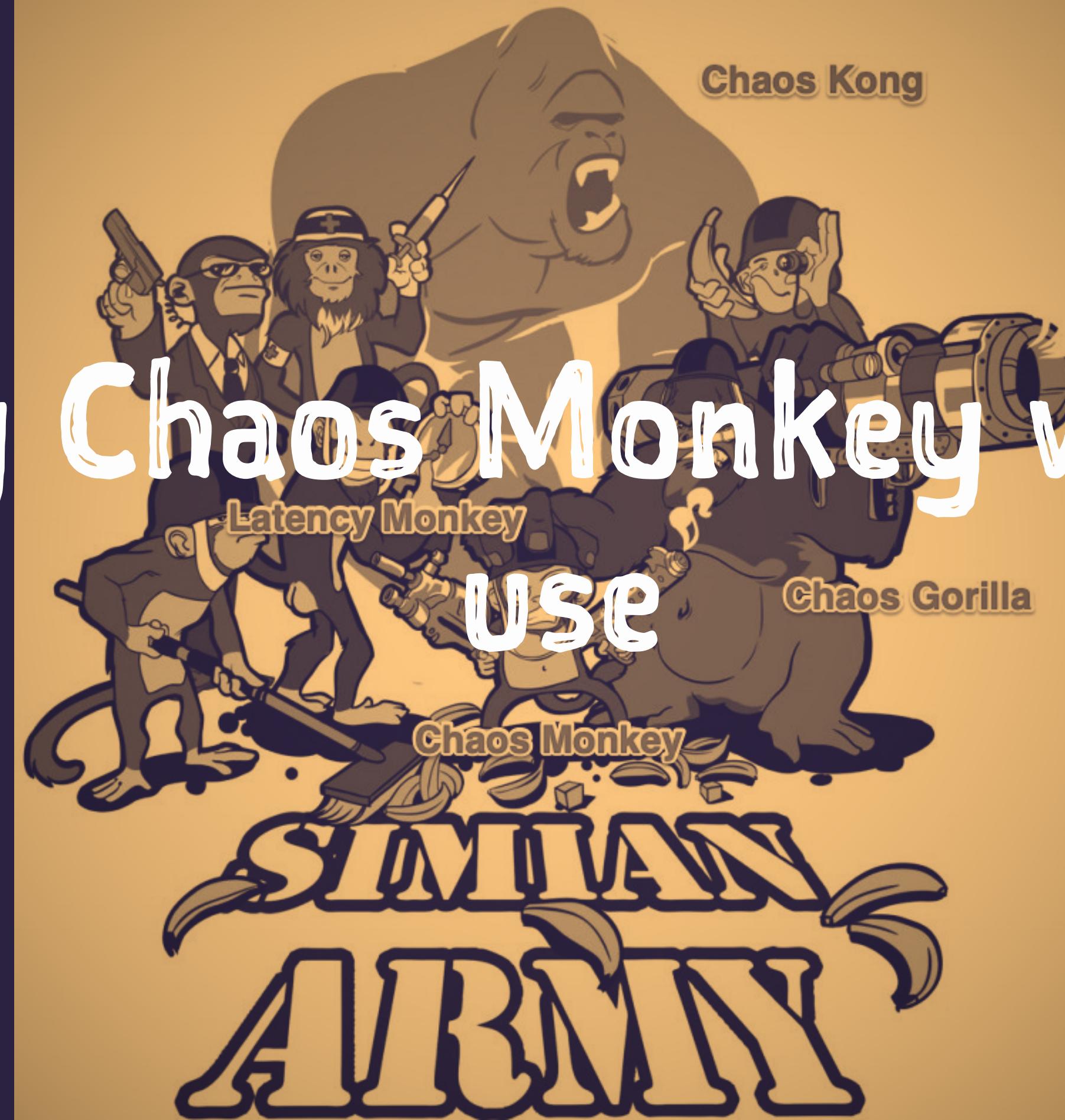
Fallbacks

Act II: Chaos at Netflix when I got there



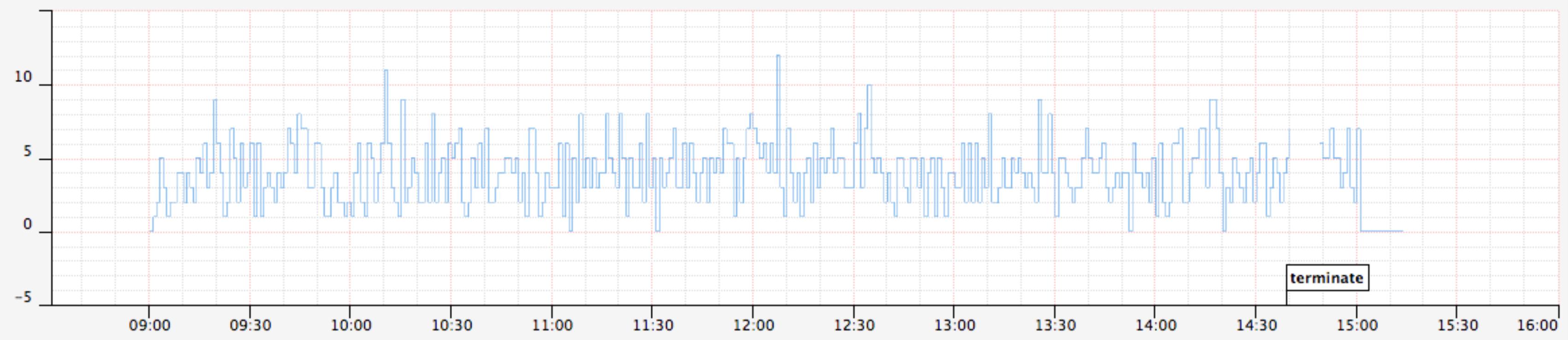


Only Chaos Monkey was in
use



Chaos Monkey randomly
terminates instances in
production

Terminations



Chaos Monkey had already
exposed single-instance
termination weaknesses

Latency monkey was too
dangerous

FIT: Failure Injection Testing

Inject failure or latency at
"injection points" in code

Example injection point:
remote procedure call

Failures are scoped, not
random

Example: Is the bookmarks service critical?

NETFLIX

Home TV Shows Movies Recently Added My List

Continue Watching for Stacy and Lorin

NETFLIX THE CROWN

the office

NETFLIX QUEER EYE more than a makeover

NETFLIX DISENCHANTMENT

Because you watched Kim's Convenience

NETFLIX ROSTERED ON

NETFLIX ALL ABOUT THE WASHINGTONS

Schitt's Creek

NETFLIX Great News

Witty TV Shows

The Good Place NEW EPISODES

New Girl

FRASIER

Cheers

Personal

Secure | https://www.netflix.com/browse

3

55

NETFLIX

Home TV Shows Movies Recently Added My List

Continue Watching for Stacy and Lorin

NETFLIX CROWN the office NETFLIX QUEER EYE more than a makeover NETFLIX DISENCHANTMENT

Because you watched Convenience

ROSTERED ON ALL ABOUT THE WASHINGTONS Schitt's Creek Great News

Witty TV Shows

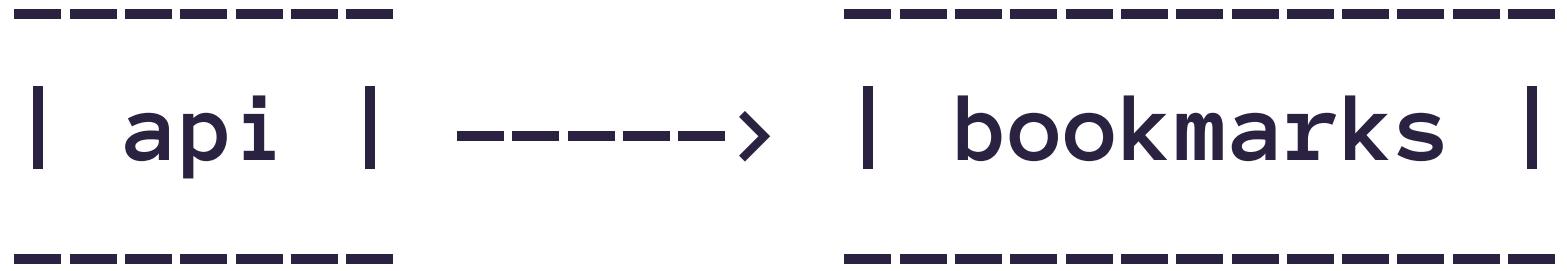
The Good Place NEW EPISODES New Girl FRASIER Cheers

Personal

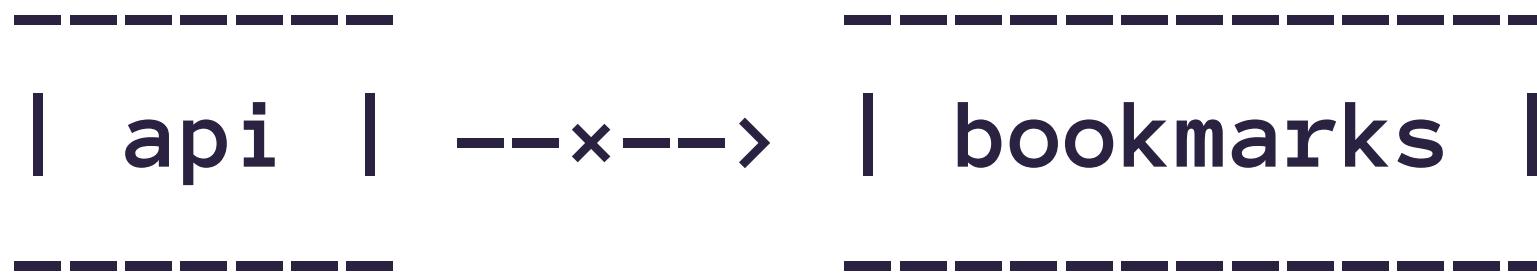
Secure | https://www.netflix.com/browse

3

56



Fail calls from the “api” service to the “bookmarks” service for account “123456”



Many service failures look
like errors or latency

Great for testing with a
single device

Some problems only appear
when many calls fail

503 Service Unavailable

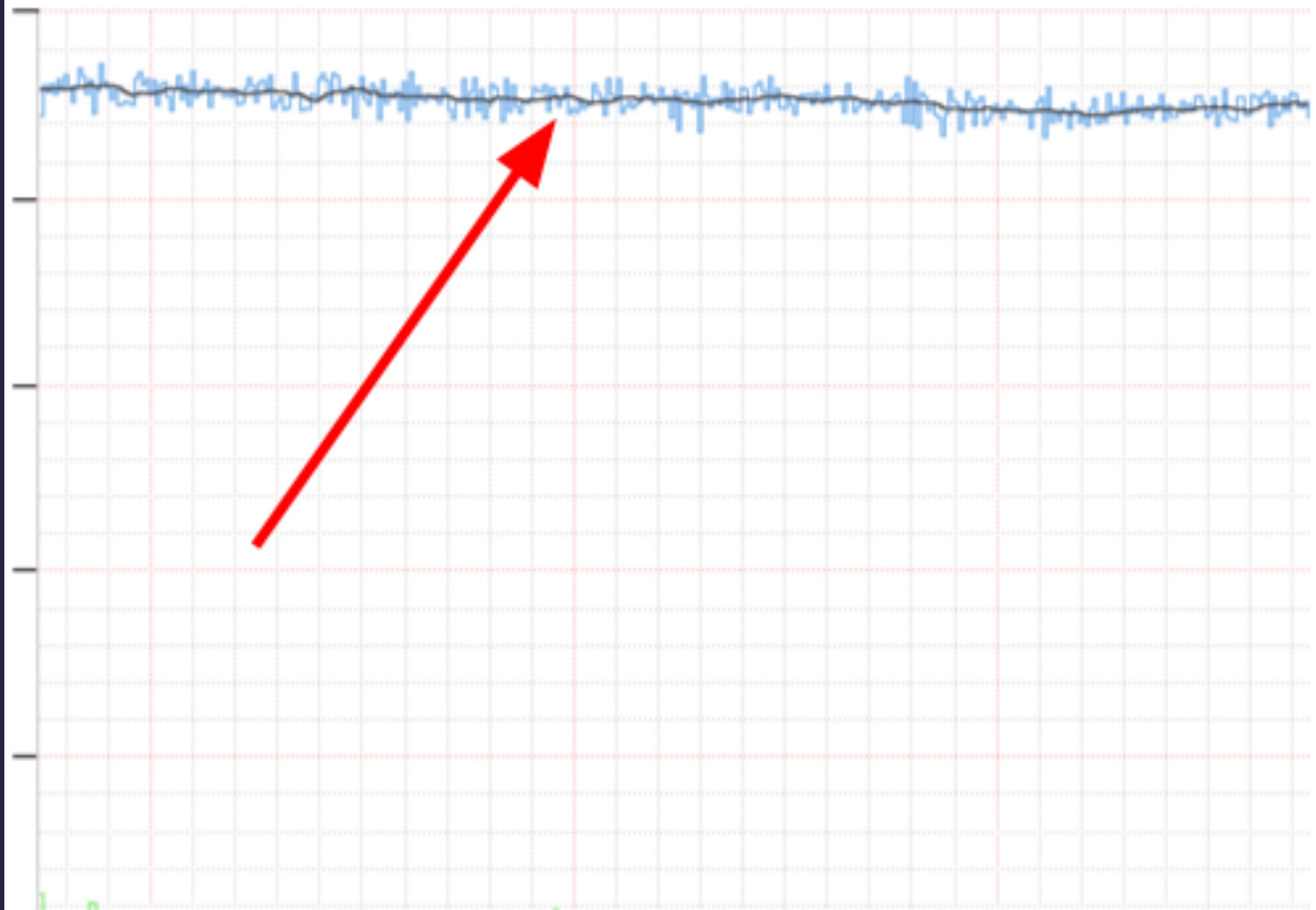
FIT supported large-scale
failure injection

Example: Inject failure for
10% of customer traffic

How much should you inject?

Too much: unnecessary
customer pain

Too little: can't tell if
there's a vulnerability





Did this have impact?

Act III: Chaos as experimentation





Chaos Automation



Want: clear signal if failure
injection having negative
impact...

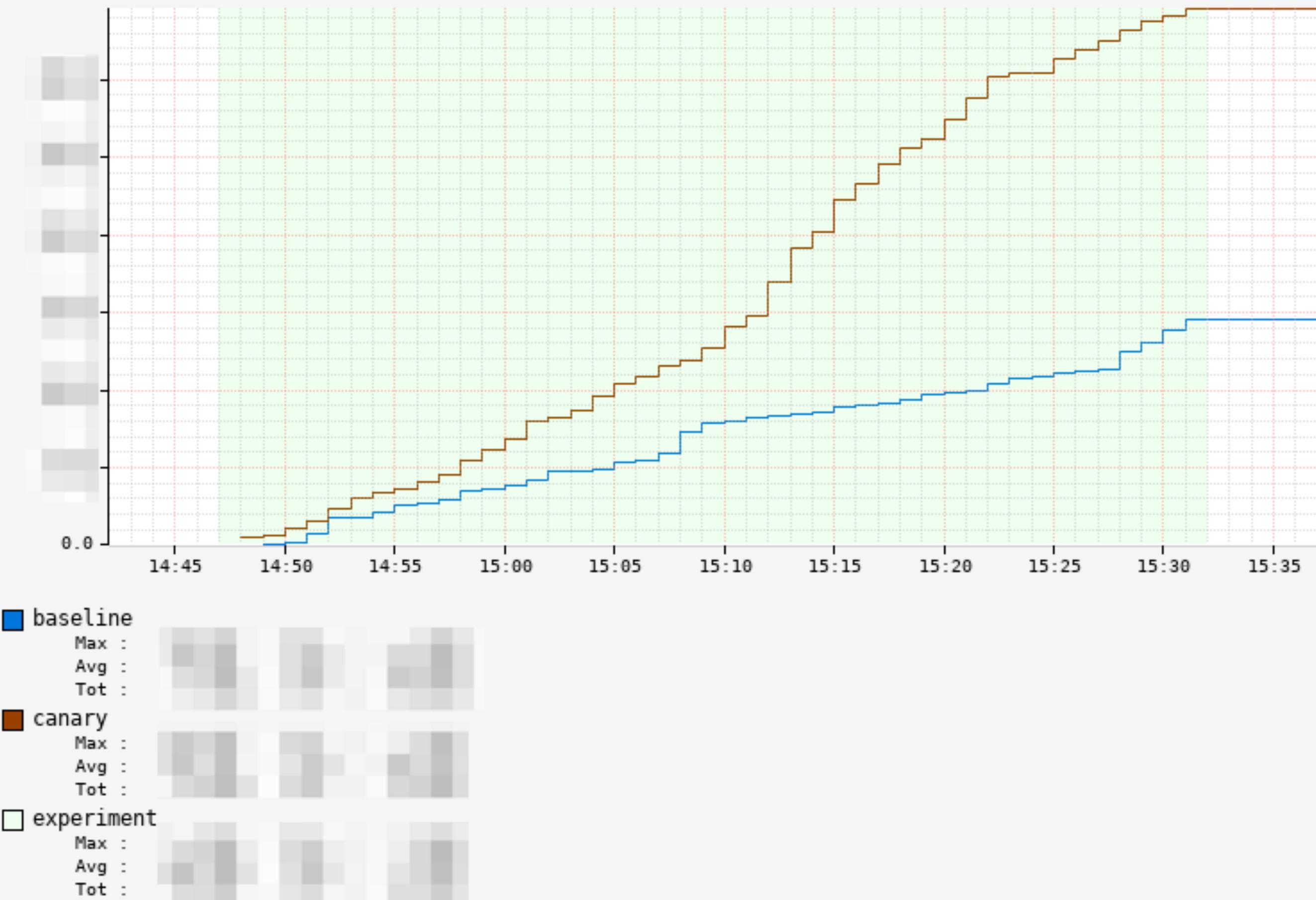
...on customers...

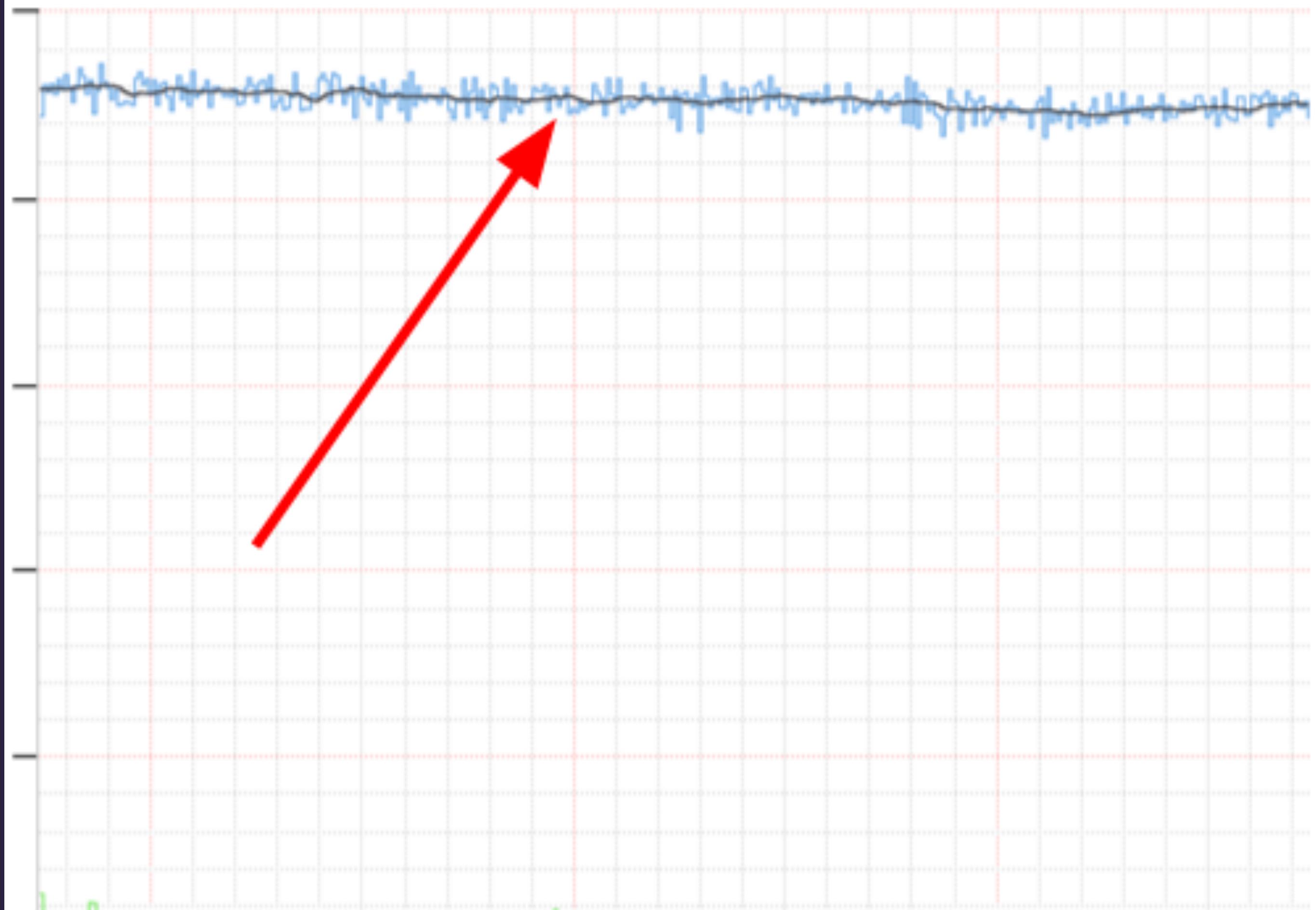
...and on services

Big idea: stickiness

Failure injection sessions
are sticky to users

SPS Errors (cumulative)

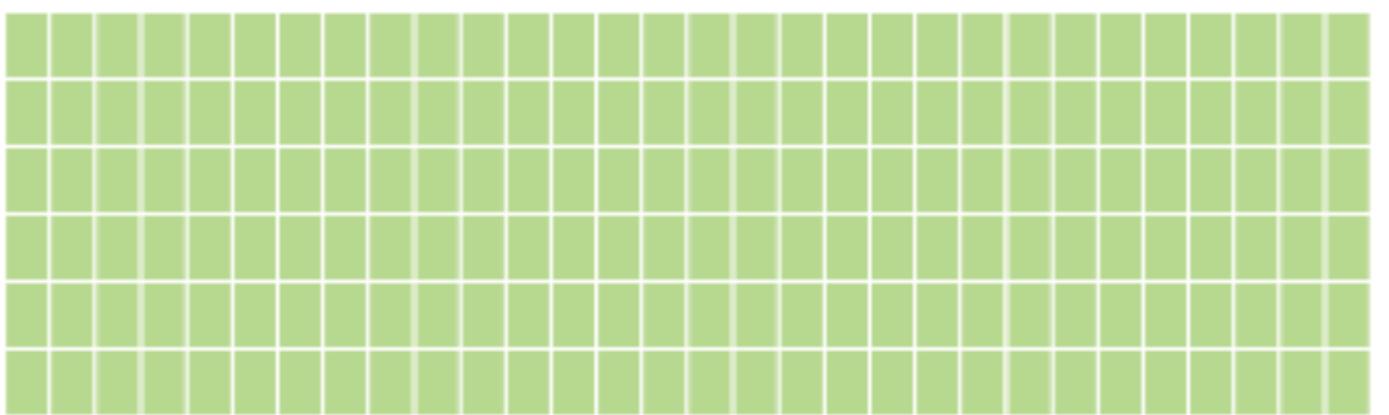




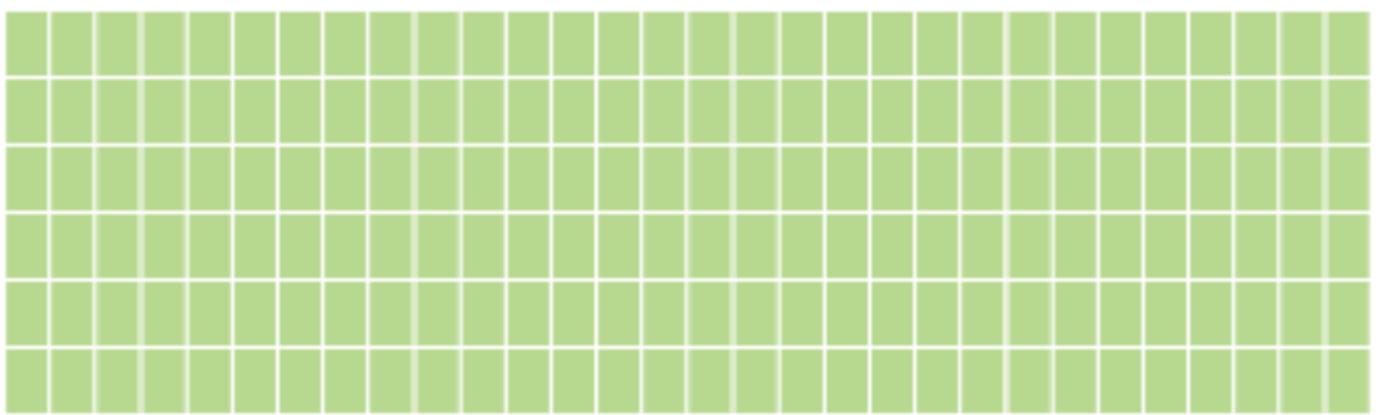
Failure injection sessions
are sticky to clusters

A

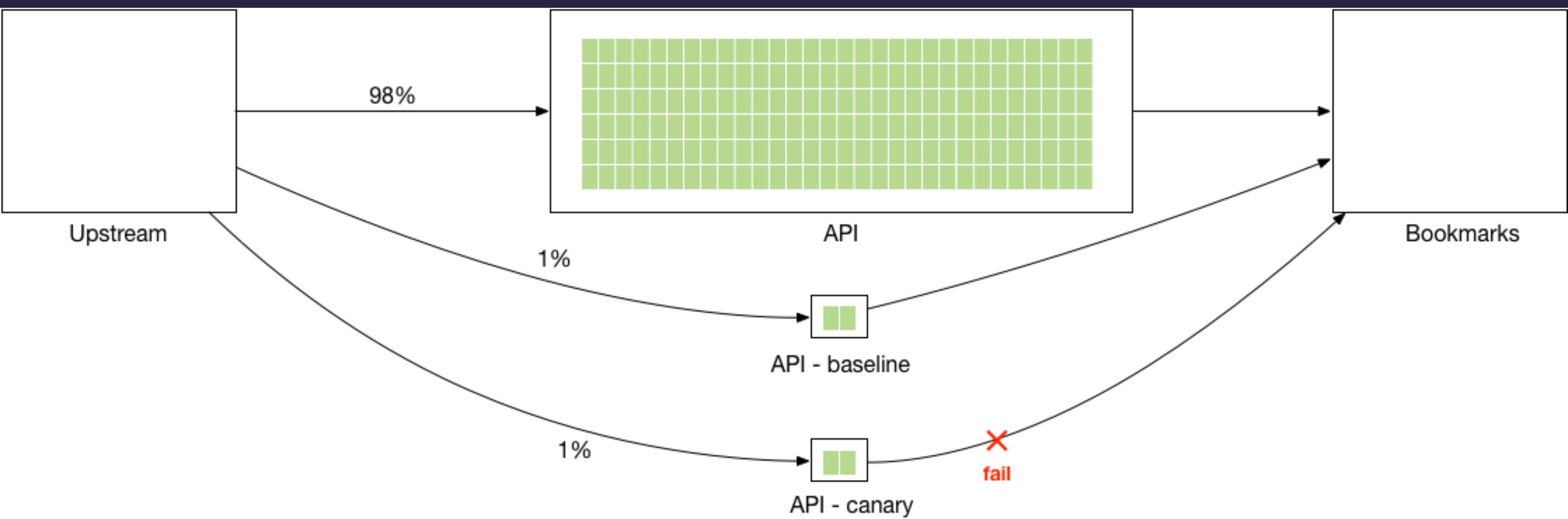
B-1

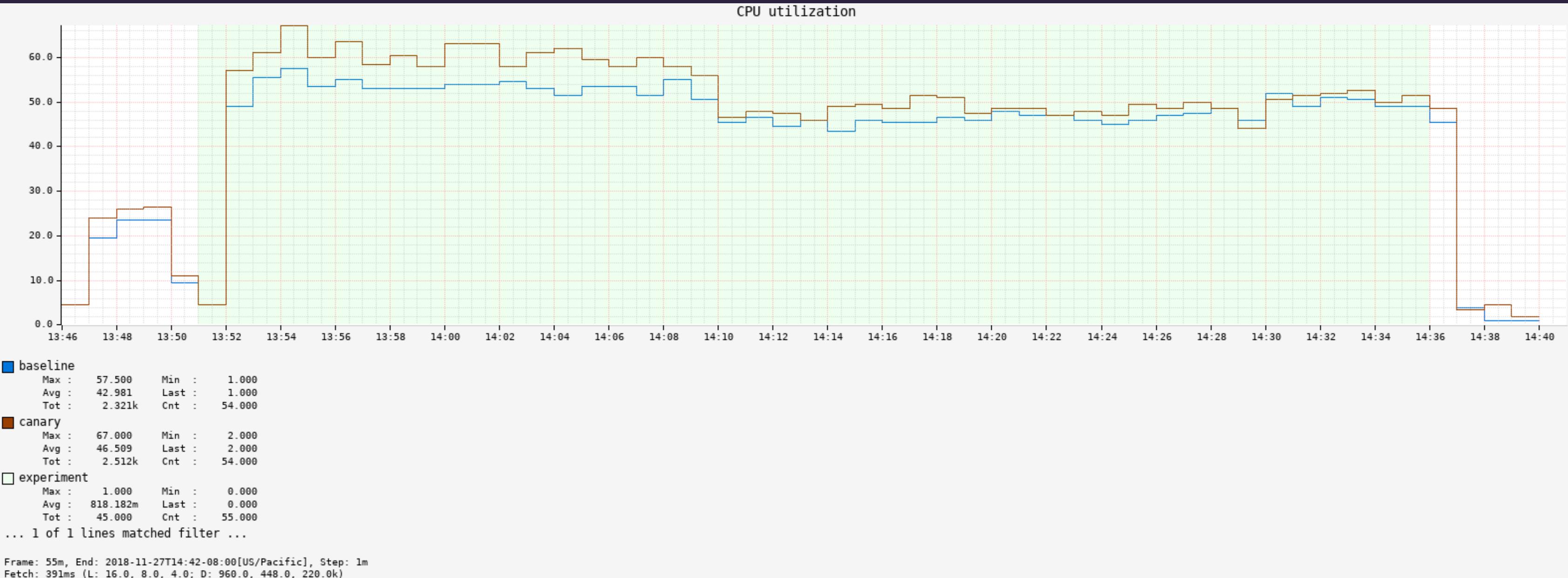


B-2



We can do controlled
experiments!





How do we do this safely?

STOP CHAOS

Automatic stop



(<5 minutes)

Business hours only

Limit number of
simultaneous runs

How do we scale this?

First attempt: self-serve

Actively engage with
multiple teams

Didn't see uptake after
engagements



Second attempt:
automatically generate
experiments

Problem: need to
understand services to
design experiments

What other services do
they communicate with?

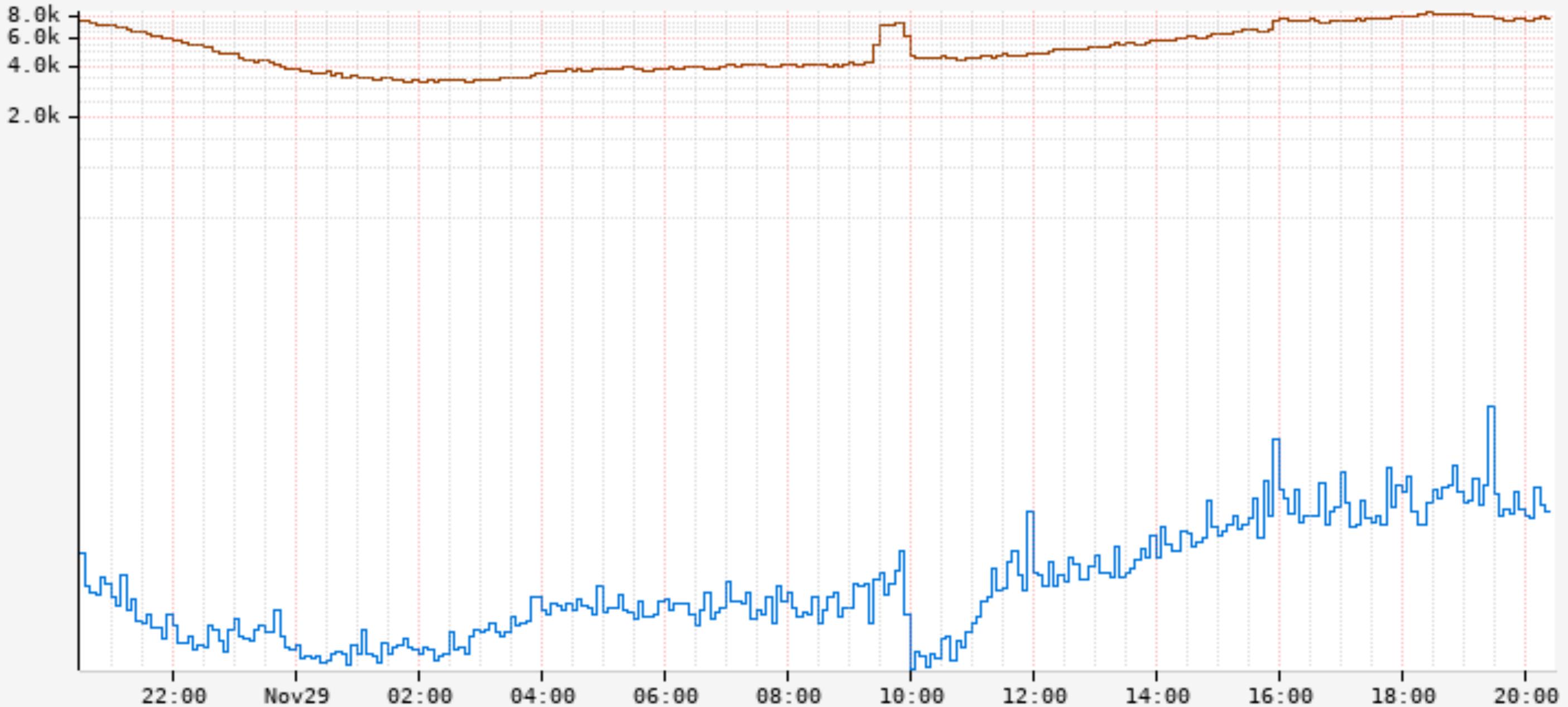


Which RPCs do we believe
are safe to fail?

Heuristics!

Is there a fallback?

Does the fallback ever get
invoked?



█ countSuccess

Max : 8.487k Min : 3.258k
Avg : 5.321k Last : 7.870k
Tot : 1.527M Cnt : 287.000

█ countFallbackSuccess

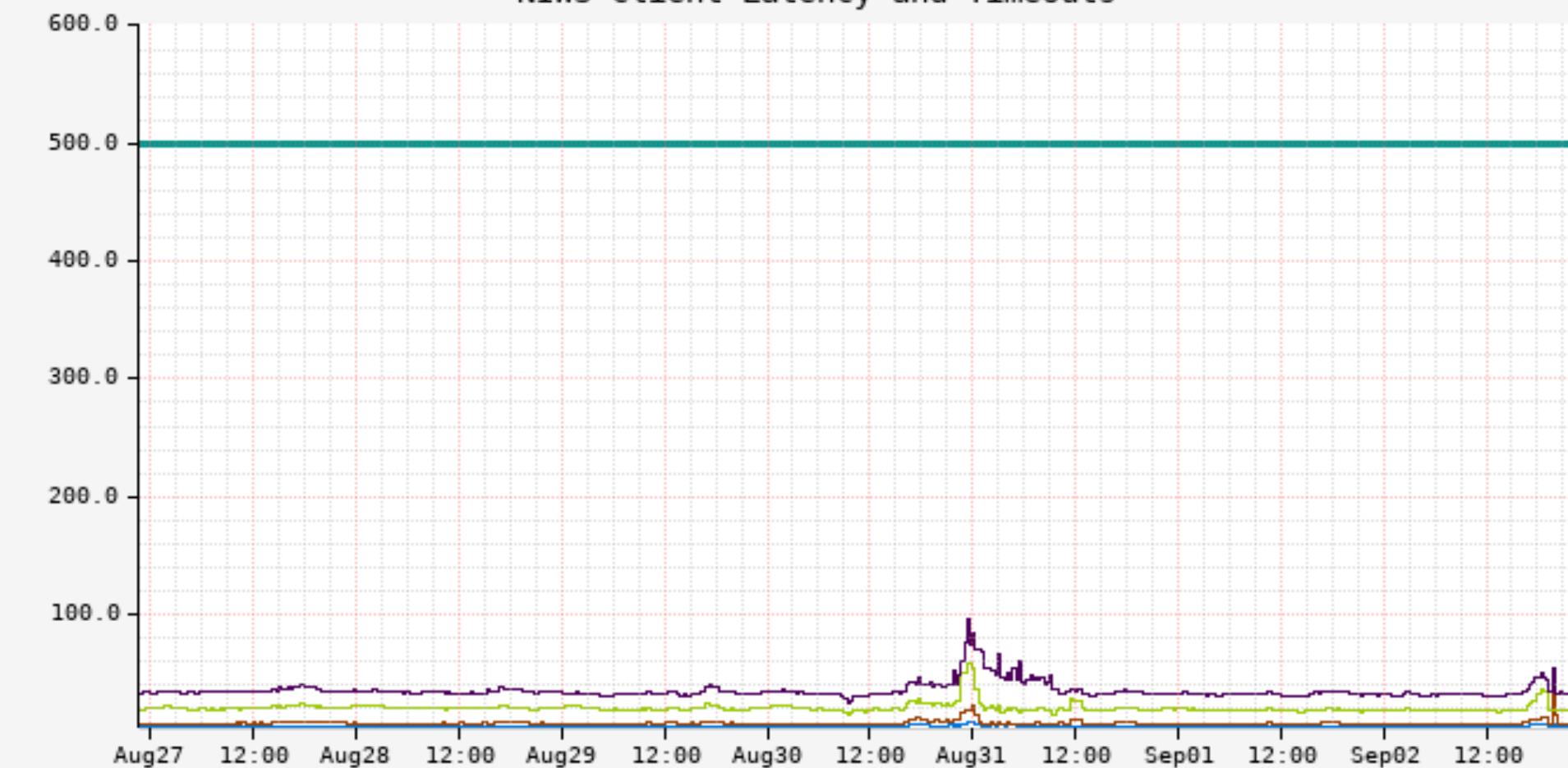
Max : 37.480 Min : 683.333m
Avg : 4.095 Last : 8.537
Tot : 1.175k Cnt : 287.000

Frame: 1d, End: 2018-11-29T20:35:08:00[US/Pacific], Step: 5m

Fetch: 928ms (L: 13.5k, 1.8k, 2.0; D: 811.1k, 531.6k, 576.0k)

How much latency should we inject?

NIWS Client Latency and Timeouts



Average Latency

Max :	8.329	Min :	3.647
Avg :	5.119	Last :	5.122
Tot :	2.575k	Cnt :	503.000

95th Percentile Latency

Max :	45.888	Min :	5.637
Avg :	7.806	Last :	7.527
Tot :	3.926k	Cnt :	503.000

99th Percentile Latency

Max :	58.698	Min :	14.730
Avg :	20.539	Last :	18.798
Tot :	10.331k	Cnt :	503.000

99.5th Percentile Latency

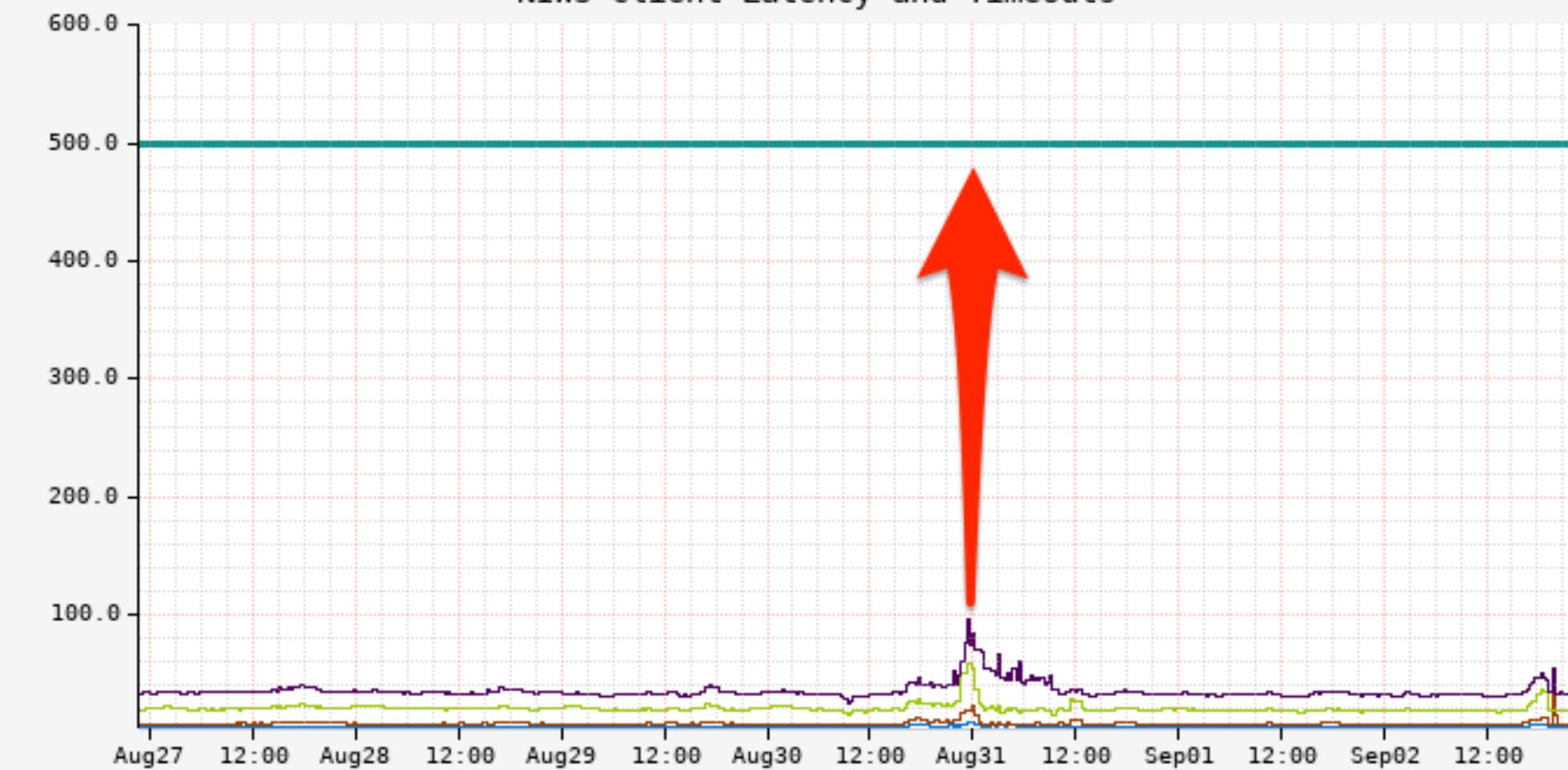
Max :	95.711	Min :	23.781
Avg :	35.247	Last :	32.656
Tot :	17.729k	Cnt :	503.000

Configured Timeout 1: 500 ms

Max :	500.000	Min :	500.000
Avg :	500.000	Last :	500.000
Tot :	252.000k	Cnt :	504.000

Frame: 1w, End: 2018-09-02T23:00:07:00[US/Pacific], Step: 20m
Fetch: 913ms (L: 242.2k, 11.8k, 5.0; D: 14.5M, 5.9M, 2.5M)

NIWS Client Latency and Timeouts



Average Latency

Max :	8.329	Min :	3.647
Avg :	5.119	Last :	5.122
Tot :	2.575k	Cnt :	503.000

95th Percentile Latency

Max :	45.888	Min :	5.637
Avg :	7.806	Last :	7.527
Tot :	3.926k	Cnt :	503.000

99th Percentile Latency

Max :	58.698	Min :	14.730
Avg :	20.539	Last :	18.798
Tot :	10.331k	Cnt :	503.000

99.5th Percentile Latency

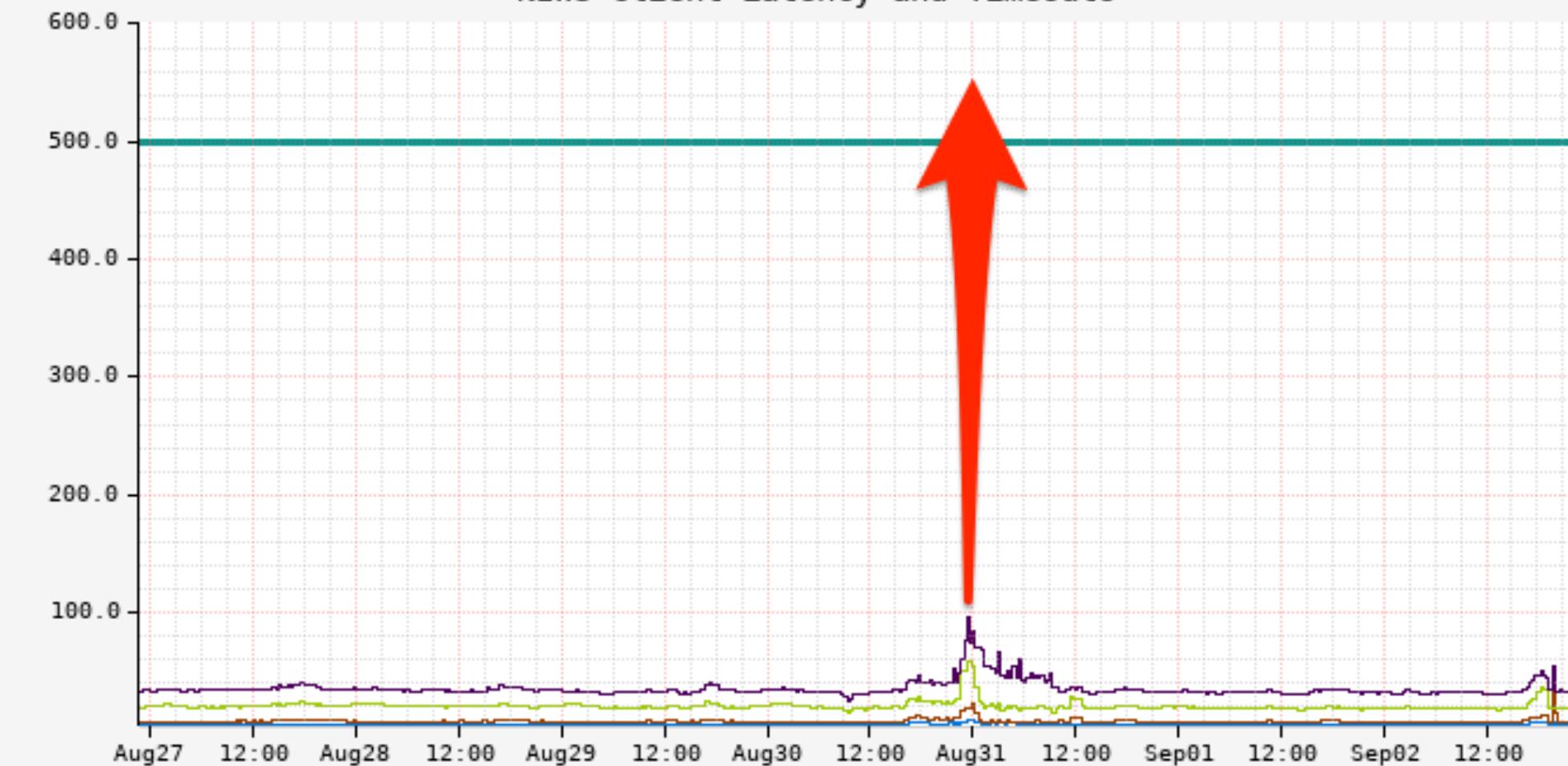
Max :	95.711	Min :	23.781
Avg :	35.247	Last :	32.656
Tot :	17.729k	Cnt :	503.000

Configured Timeout 1: 500 ms

Max :	500.000	Min :	500.000
Avg :	500.000	Last :	500.000
Tot :	252.000k	Cnt :	504.000

Frame: 1w, End: 2018-09-02T23:00:07:00[US/Pacific], Step: 20m
Fetch: 913ms (L: 242.2k, 11.8k, 5.0; D: 14.5M, 5.9M, 2.5M)

NIWS Client Latency and Timeouts



Average Latency

Max :	8.329	Min :	3.647
Avg :	5.119	Last :	5.122
Tot :	2.575k	Cnt :	503.000

95th Percentile Latency

Max :	45.888	Min :	5.637
Avg :	7.806	Last :	7.527
Tot :	3.926k	Cnt :	503.000

99th Percentile Latency

Max :	58.698	Min :	14.730
Avg :	20.539	Last :	18.798
Tot :	10.331k	Cnt :	503.000

99.5th Percentile Latency

Max :	95.711	Min :	23.781
Avg :	35.247	Last :	32.656
Tot :	17.729k	Cnt :	503.000

Configured Timeout 1: 500 ms

Max :	500.000	Min :	500.000
Avg :	500.000	Last :	500.000
Tot :	252.000k	Cnt :	504.000

Frame: 1w, End: 2018-09-02T23:00:07:00[US/Pacific], Step: 20m
Fetch: 913ms (L: 242.2k, 11.8k, 5.0; D: 14.5M, 5.9M, 2.5M)

We found vulnerabilities!

Still requires human effort
to interpret results

Experimental design limited
by our heuristics

Current state: hybrid
approach

Busy season: right before
the holidays

Act IV: Lessons learned

Safety

It needs to be safe, or
nobody will use it

Safe = limited impact



Simplicity is prerequisite for reliability

-- Edsger Dijkstra



Safety adds complexity

$\text{WithinLimit} \triangleq \text{Sum}(\text{running}) \leq \text{TrafficLimit}$
 $\text{TypeOK} \triangleq \wedge \text{queue} \in \text{SUBSET Runs}$
 $\wedge \text{owned} \in \text{SUBSET Runs}$
 $\wedge \text{running} \in \text{SUBSET Runs}$
 $\wedge \text{traffic} \in [\text{Runs} \rightarrow \text{Nat} \setminus \{0\}]$
 $\wedge \text{candidate} \in [\text{ProcSet} \rightarrow \text{Runs} \cup \{\text{NoRun}\}]$
 $\wedge \text{known} \in [\text{ProcSet} \rightarrow \text{SUBSET Runs}]$
 $\wedge \text{pc} \in [\text{ProcSet} \rightarrow \{\text{"p1"}, \text{"p2"}, \text{"p3"}, \text{"Done"}\}]$
 $\text{Inv} \triangleq \wedge \text{TypeOK}$
 $\wedge \forall i \in \text{ProcSet} : \text{known}[i] \subseteq \text{owned}$
 $\wedge \forall i \in \text{ProcSet} : \vee \text{candidate}[i] = \text{NoRun}$
 $\quad \vee \text{candidate}[i] \in \text{owned}$
 $\wedge \forall \text{run} \in \text{running} : \exists i \in \text{ProcSet} : \wedge \text{pc}[i] = \text{"Done"}$
 $\quad \quad \quad \wedge \text{candidate}[i] = \text{run}$
 $\wedge \forall i, j \in \text{ProcSet} : \vee \text{known}[i] \subseteq \text{known}[j]$
 $\quad \quad \quad \vee \text{known}[j] \subseteq \text{known}[i]$
 $\wedge \text{WithinLimit}$
ASSUME $\text{NumWorkersInNat} \triangleq \text{NumWorkers} \in \text{Nat} \setminus \{0\}$
ASSUME $\text{TrafficLimitInNat} \triangleq \text{TrafficLimit} \in \text{Nat} \setminus \{0\}$
LEMMA $\text{SumPrime} \triangleq \forall S \in \text{SUBSET Runs} : (\text{Sum}(S))' = \text{Sum}(S')$
LEMMA $\text{EmptySumIsZero} \triangleq \text{Sum}(\{\}) = 0$
LEMMA $\text{SumInNat} \triangleq \forall S \in \text{SUBSET Runs} : \text{Sum}(S) \in \text{Nat}$
THEOREM $\text{Spec} \Rightarrow \square \text{WithinLimit}$
(1) USE DEF ProcSet, Inv
(1)1. Init \Rightarrow Inv
(2) SUFFICES ASSUME Init
PROVE Inv
OBVIOUS
(2)1. TypeOK
BY DEF TypeOK
(2)2. $\forall i \in \text{ProcSet} : \text{known}[i] \subseteq \text{owned}$
OBVIOUS
(2)3. $\forall i \in \text{ProcSet} : \vee \text{candidate}[i] = \text{NoRun}$
 $\quad \vee \text{candidate}[i] \in \text{owned}$

You better have damn good
tests around your failure
injection logic...

...especially if it's a shared
library in every app!

```
18:18:00,094 ERROR FitContextImpl:195 - Fit Error checking or injecting failure
java.lang.NullPointerException
    at com.netflix.fit.InjectionPointImpl.wildcardMatch(InjectionPointImpl.java:133)
    at com.netflix.fit.scenario.FitScenarioImpl.shouldImpact(FitScenarioImpl.java:45)
    at com.netflix.fit.FitContextImpl.shouldInjectFailure(FitContextImpl.java:130)
    at com.netflix.fit.FitContextImpl.checkAndInjectFailure(FitContextImpl.java:191)
    at com.netflix.fit.FitContext.checkAndInjectFailure(FitContext.java:40)
    at com.netflix.server.base.fit.FitHandler.handle(FitHandler.java:34)
    at com.netflix.server.base.NFFilter.safeDoFilter(NFFilter.java:574)
    at com.netflix.server.base.NFFilter.access$200(NFFilter.java:234)
    at com.netflix.server.base.NFFilter$3.call(NFFilter.java:482)
    at com.netflix.server.base.NFFilter$3.call(NFFilter.java:479)
    at com.netflix.lang.BindingContexts.callWithNewContext(BindingContexts.java:182)
    at com.netflix.server.base.NFFilter.doFilter(NFFilter.java:479)
    at com.google.inject.servlet.FilterChainInvocation.doFilter(FilterChainInvocation.java:82)
    at com.google.inject.servlet.ManagedFilterPipeline.dispatch(ManagedFilterPipeline.java:120)
    at com.google.inject.servlet.GuiceFilter.doFilter(GuiceFilter.java:135)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
    at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)
    at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:106)
    at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:502)
    at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)
```

ChAP isn't a "black box"

Experimental design is a
skill

Work isn't done when
automated experiment
reveals a weakness

Confirm it's a genuine
problem

Communicate effectively
back to service owners

Lots of tuning required

Length of experiment

Amount of traffic impacted

Auto-stop thresholds

Error counts are noisy

Leverage your internal
tooling ecosystem

ChAP is really an orchestration tool

- Fault injection
- Sticky routing
- Continuous deployment
- Tracing
- Telemetry
- Automated canary analysis

The more heterogeneous
your ecosystem, the harder
life will be

Java -> Node.js

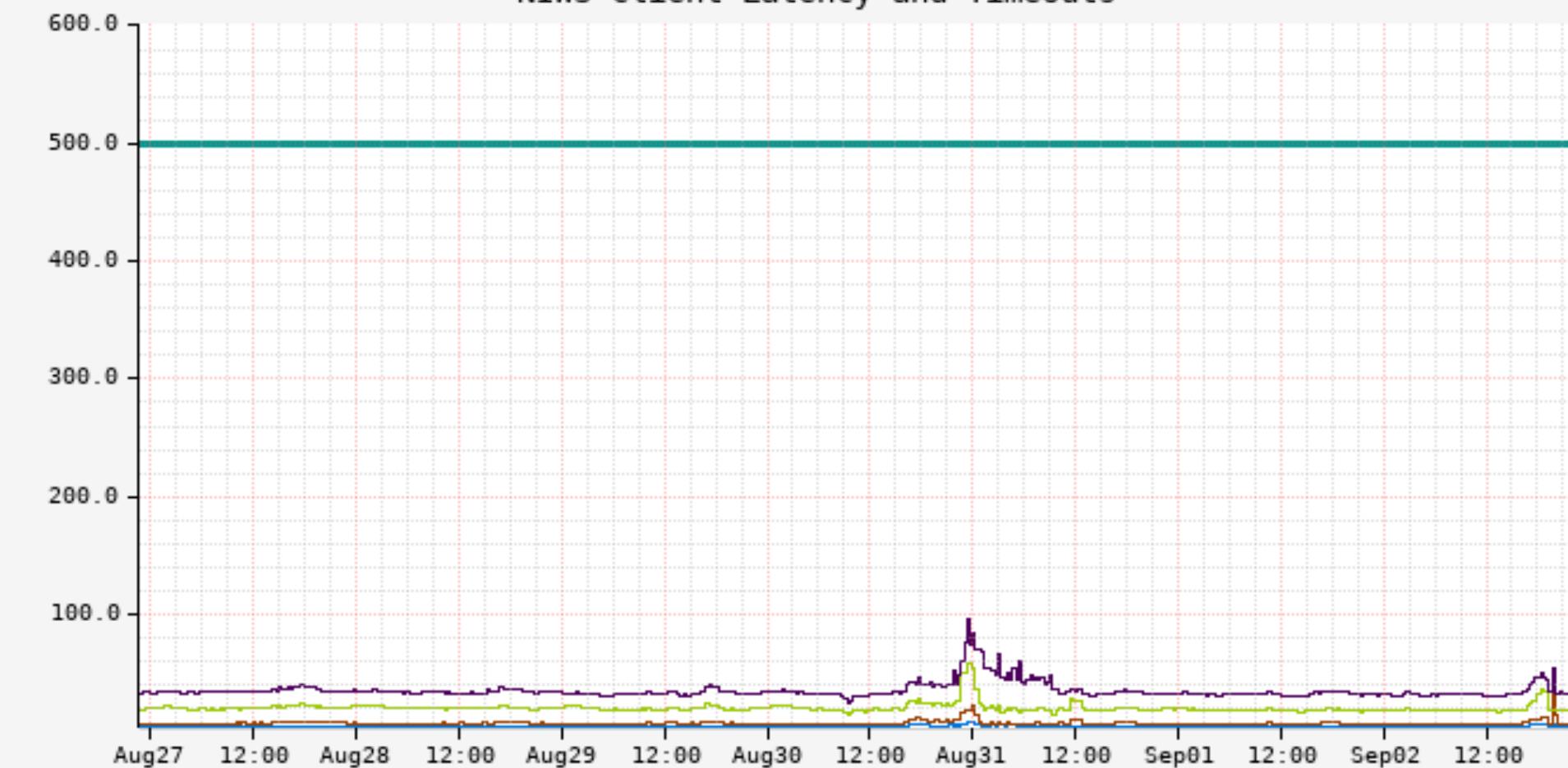
REST -> gRPC

VMs -> containers

Unexpected benefits

Info for experiment
generation was useful to
service owners

NIWS Client Latency and Timeouts



Average Latency

Max :	8.329	Min :	3.647
Avg :	5.119	Last :	5.122
Tot :	2.575k	Cnt :	503.000

95th Percentile Latency

Max :	45.888	Min :	5.637
Avg :	7.806	Last :	7.527
Tot :	3.926k	Cnt :	503.000

99th Percentile Latency

Max :	58.698	Min :	14.730
Avg :	20.539	Last :	18.798
Tot :	10.331k	Cnt :	503.000

99.5th Percentile Latency

Max :	95.711	Min :	23.781
Avg :	35.247	Last :	32.656
Tot :	17.729k	Cnt :	503.000

Configured Timeout 1: 500 ms

Max :	500.000	Min :	500.000
Avg :	500.000	Last :	500.000
Tot :	252.000k	Cnt :	504.000

Frame: 1w, End: 2018-09-02T23:00:07:00[US/Pacific], Step: 20m
Fetch: 913ms (L: 242.2k, 11.8k, 5.0; D: 14.5M, 5.9M, 2.5M)

Engineers created new use
cases (sticky canary)

SPS Errors (cumulative)

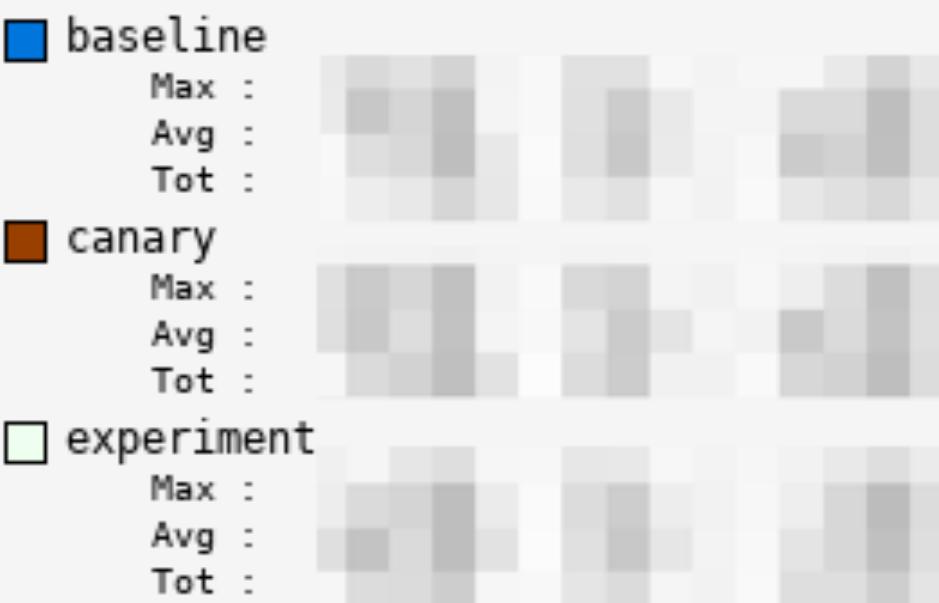
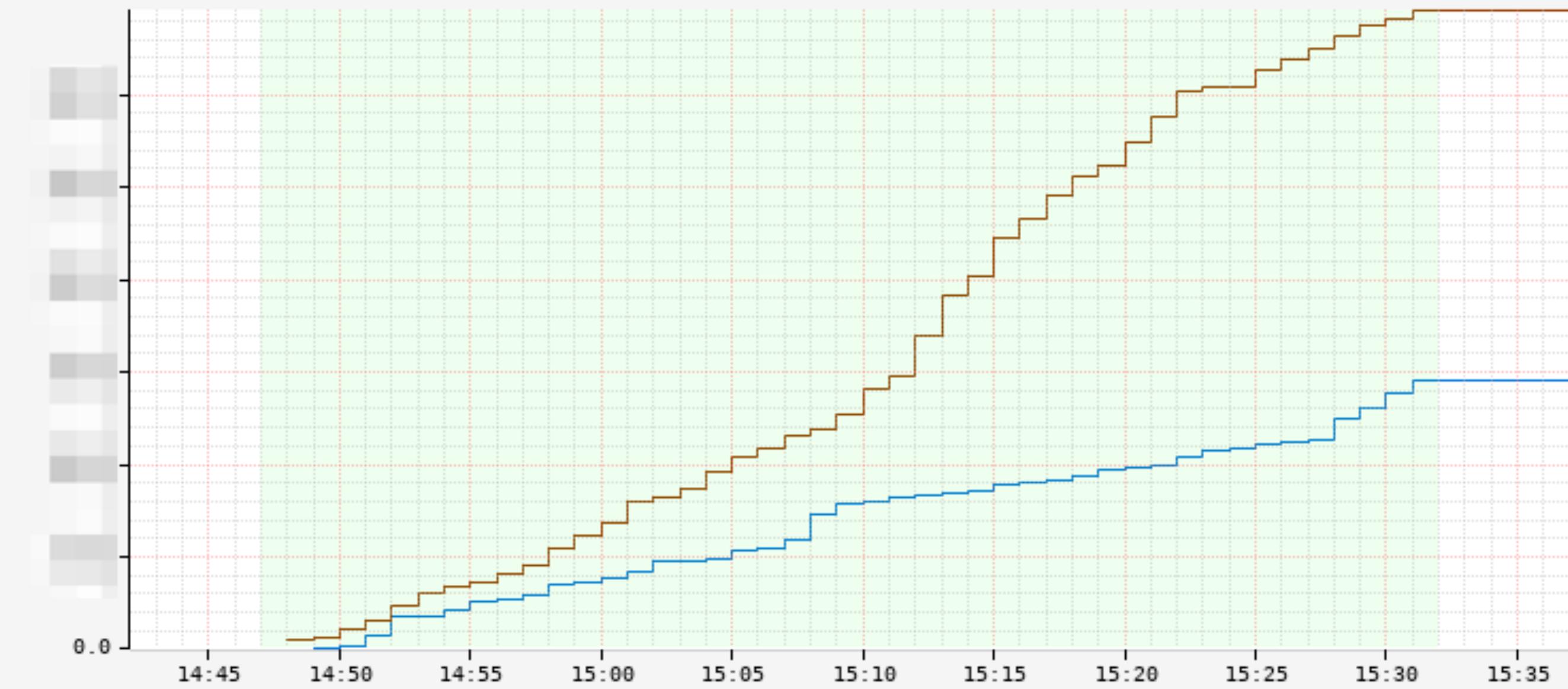


Image credits

- Knobs, Ian Harding, CC-BY-NC-SA 2.0: <https://flic.kr/p/d9sCZ>
- Lightning Storm, Benjamen Benson, CC BY CA 2.0: <https://flic.kr/p/aesuRd>
- Portrait of Edsger W. Dijkstra, Hamilton Richards, CC BY-SA 3.0: https://en.wikipedia.org/wiki/Edsger_W._Dijkstra#/media/