What I've learned doing chaos at Netflix

Lorin Hochstein (@lhochstein)

Chaos engineering should not be endorsed by the ICSE community. Accepting a workshop pretty much endorses the topic.

-- Reviewer, rejected ICSE'16 chaos workshop proposal

Some context about Netflix

We care about availability

Whoops, something went wrong…

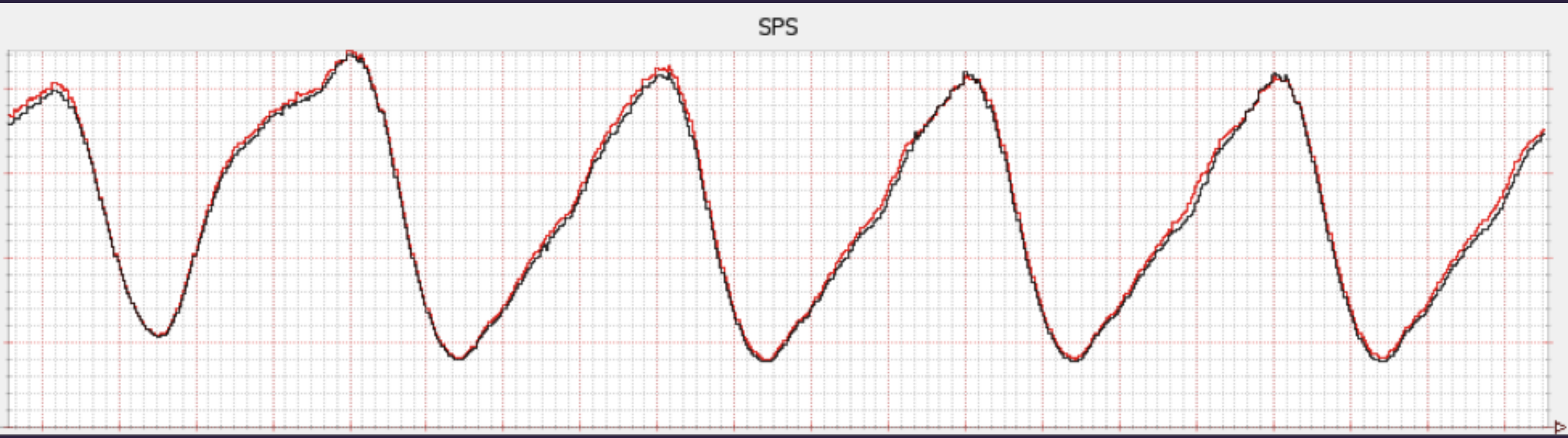
Netflix Streaming Error

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

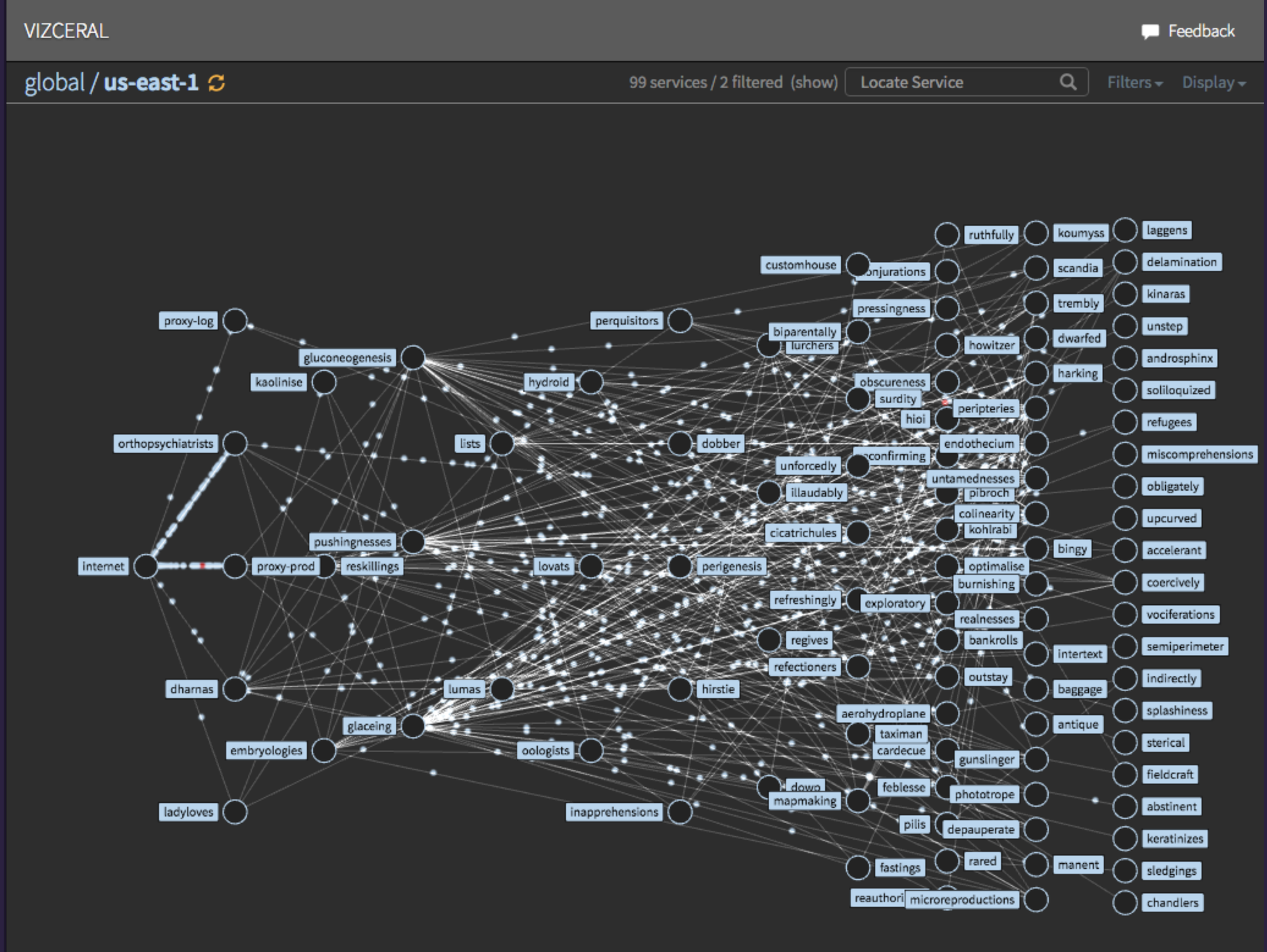
SPS: Stream starts Per Second

Number of people who hit the "play" button and successfully started

99.95%


SPS

Microservice architecture

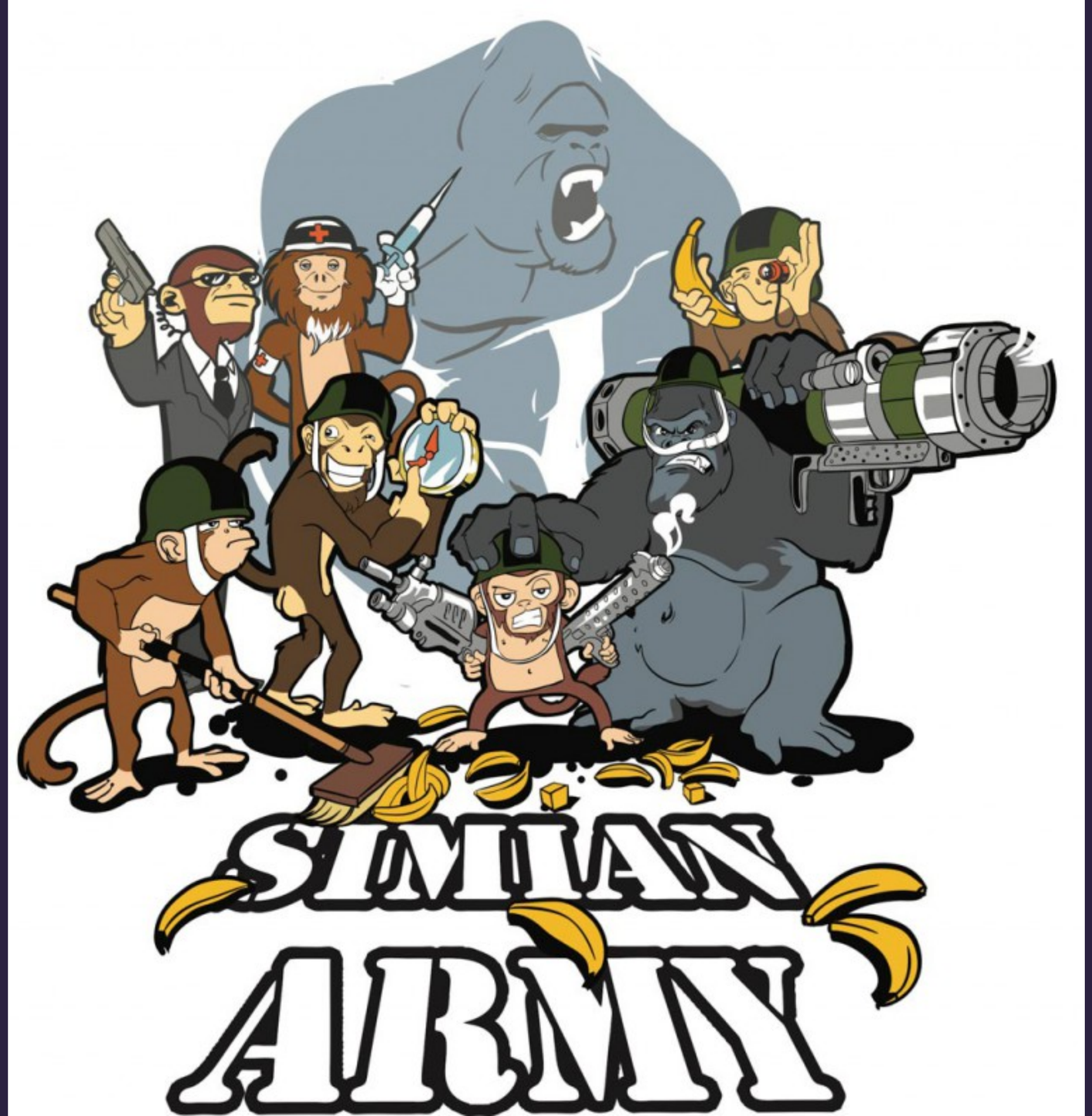

VIZCERAL
Feedback
global / us-east-1
99 services / 2 filtered (show)
Locate Service
Filters
Display
proxy-log
gluconeogenesis
kaolinise
orthopsychiatrists
internet
proxy-prod
reskillings
pushingnesses
dharnas
embryologies
ladyloves
glaceing
perquisitors
hydroid
lists
dobber
lovats
perigenesis
lumas
hirstie
oologists
inapprehensions
customhouse
biparentally
lurchers
pressingness
obscureness
surdity
hioi
unforcedly
illaudably
cicatrichules
refreshingly
regives
refectioners
aerohydroplane
taximan
cardecue
feblesse
mapmaking
pilis
fastings
reauthori
ruthfully
howitzer
peripteries
endothecium
untamednesses
pibroch
colinearity
kohlrabi
optimalise
burnishing
exploratory
realnesses
bankrolls
outstay
gunslinger
phototrope
depauperate
rared
microreproductions
koumyss
scandia
trembly
dwarfed
harking
bingy
intertext
baggage
antique
manent
laggens
delamination
kinaras
unstep
androsphinx
soliloquized
refugees
miscomprehensions
obligately
upcurved
accelerant
coercively
vociferations
semiperimeter
indirectly
splashiness
sterical
fieldcraft
abstinent
keratinizes
sledgings
chandlers

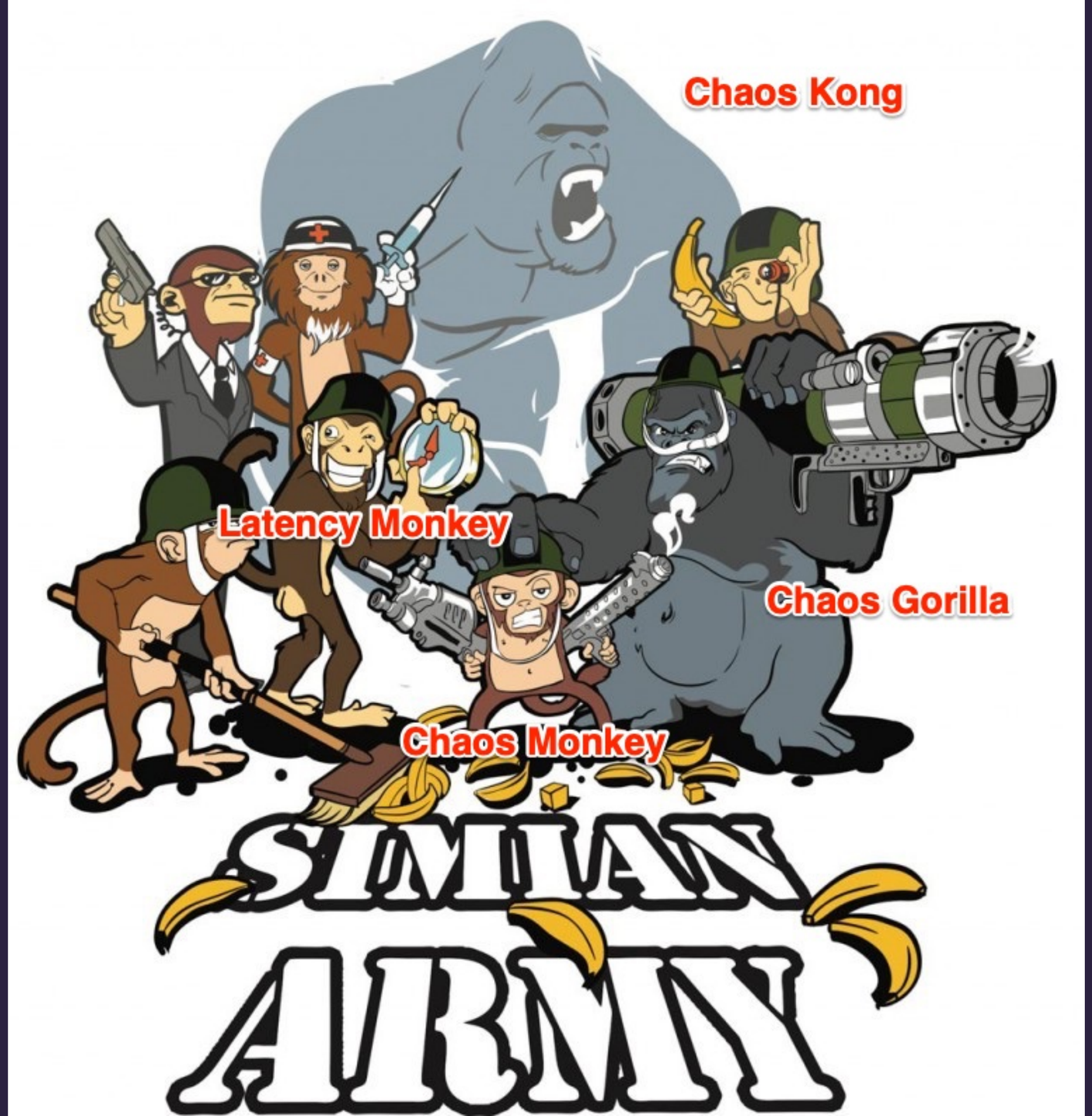

VIZCERAL
Feedback
global / us-east-1
99 services / 2 filtered (show)
Locate Service
Filters
Display
Many points of failure!
proxy-log
gluconeogenesis
kaolinise
perquisitors
hydroid
customhouse
biparentally
lurchers
pressingness
ruthfully
koumyss
scandia
trembly
dwarfed
harking
howitzer
laggens
delamination
kinaras
unstep
androsphinx
soliloquized
illaudably
pibroch
colinearity
kohlrabi
upcurved
accelerant
coercively
vociferations
semiperimeter
indirectly
splashiness
sterical
fieldcraft
abstinent
keratinizes
sledgings
chandlers
pushingnesses
internet
proxy-prod
reskillings
lovats
perigenesis
cicatrichules
bingy
optimalise
burnishing
realnesses
bankrolls
outstay
intertext
baggage
antique
refreshingly
exploratory
regives
refectioners
dharnas
lumas
hirstie
glaceing
embryologies
oologists
ladyloves
inapprehensions
aerohydroplane
taximan
cardecue
gunslinger
feblesse
mapmaking
phototrope
pilis
depauperate
rared
manent
fastings
reauthori
microreproductions

A play in three acts

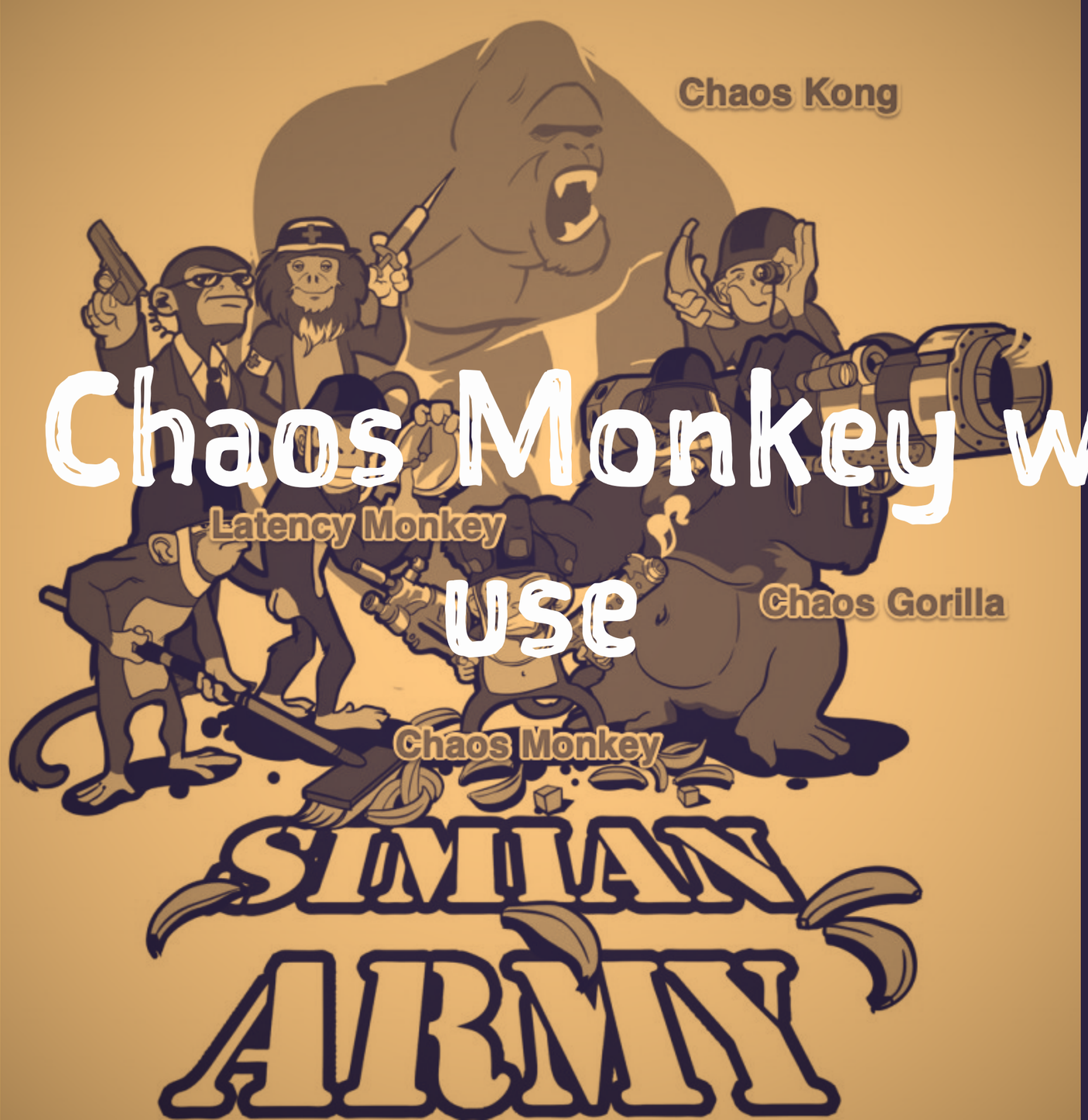
- Act I: Chaos at Netflix when I got there
- Act II: Chaos as experimentation
- Act III: Lessons learned

Act 1: Chaos at Netflix when I got there

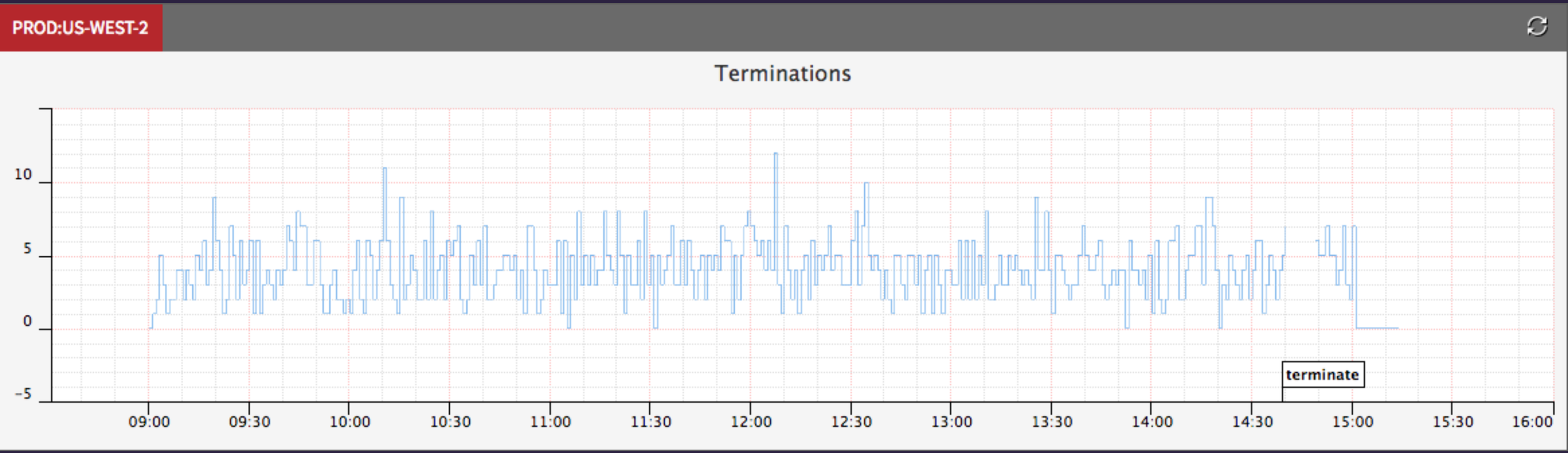

SIMIAN
ARMY


Chaos Kong
Latency Monkey
Chaos Gorilla
Chaos Monkey
SIMIAN
ARMY

Only Chaos Monkey was in use



Chaos Monkey randomly terminates instances in production


PROD:US-WEST-2
Terminations
10
5
0
-5
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
15:30
16:00
terminate

Chaos Monkey had already exposed single-instance termination weaknesses

Latency monkey was too dangerous

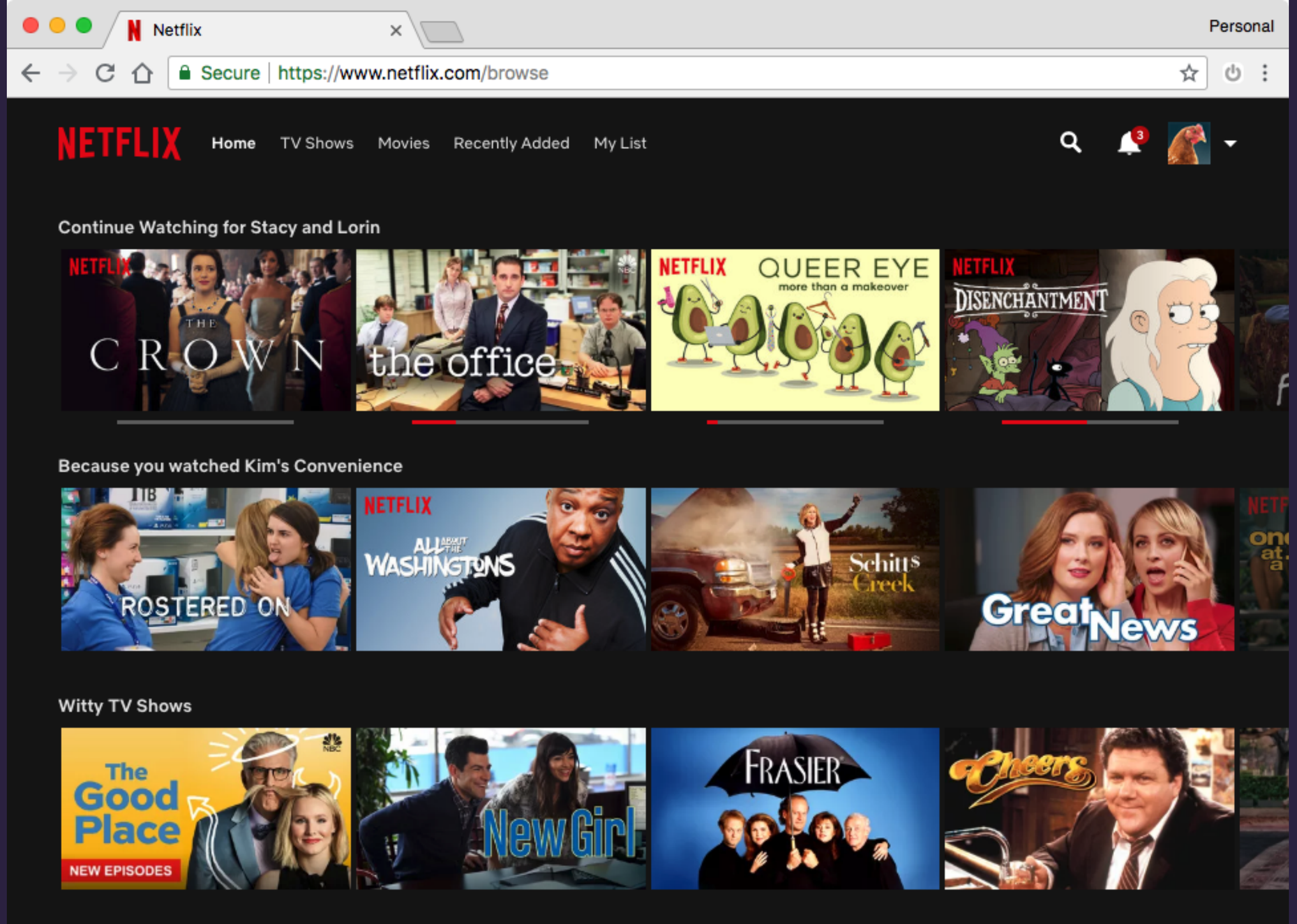
FIT: Failure Injection Testing

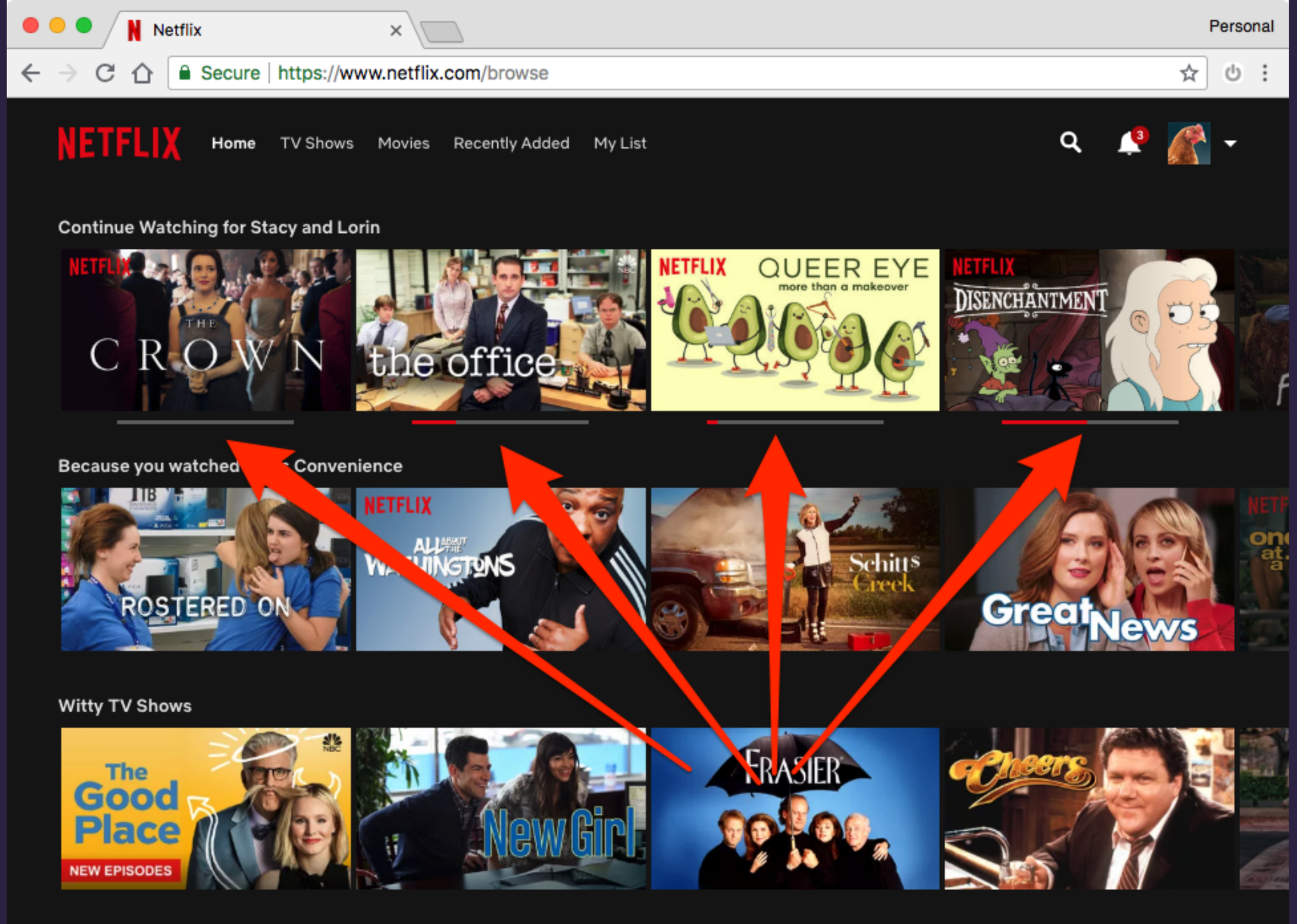
Inject failure or latency at "injection points" in code

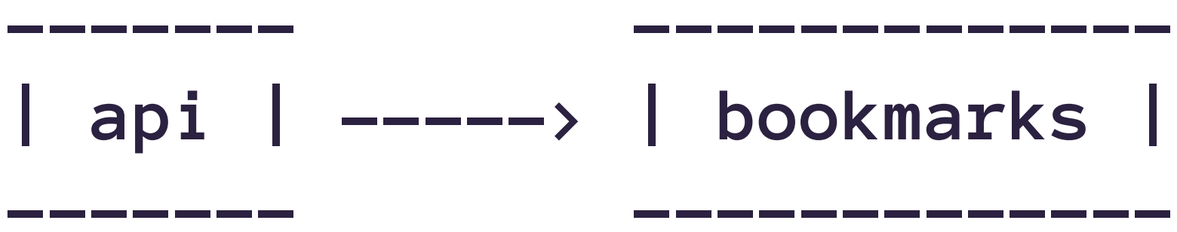
Example injection point: remote procedure call

Failures are scoped, not random

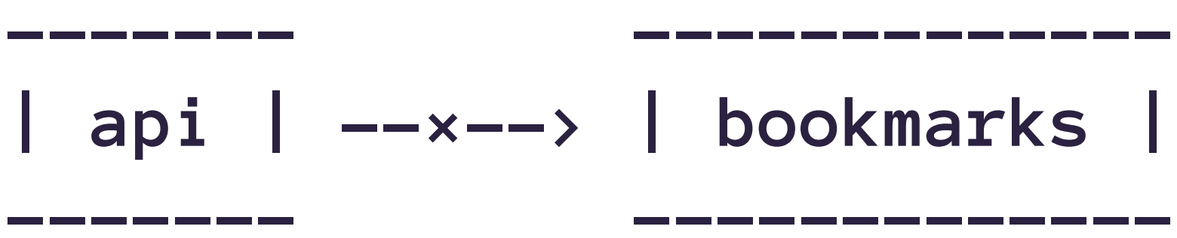
Example: Is the bookmarks service critical?


Netflix
Personal
Secure | https://www.netflix.com/browse
NETFLIX
Home
TV Shows
Movies
Recently Added
My List
Continue Watching for Stacy and Lorin
THE CROWN
the office
QUEER EYE
more than a makeover
DISENCHANTMENT
Because you watched Kim's Convenience
ROSTERED ON
ALL ABOUT THE WASHINGTONS
Schitt's Creek
Great News
Witty TV Shows
The Good Place
NEW EPISODES
New Girl
FRASIER
Cheers


Netflix
Personal
Secure | https://www.netflix.com/browse
NETFLIX
Home
TV Shows
Movies
Recently Added
My List
Continue Watching for Stacy and Lorin
THE CROWN
the office
QUEER EYE
more than a makeover
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


api
bookmarks

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

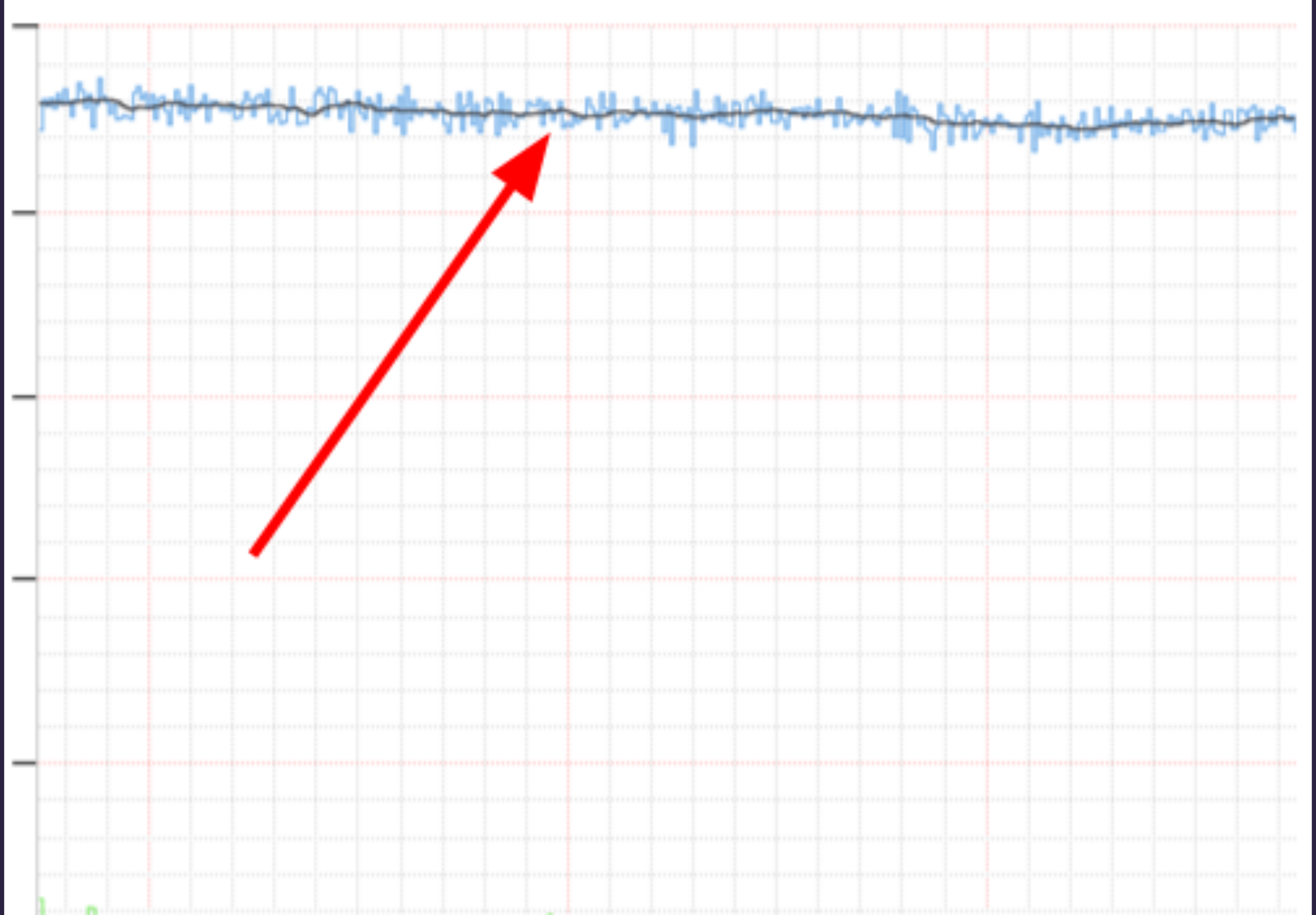
FIT supported large-scale failure injection

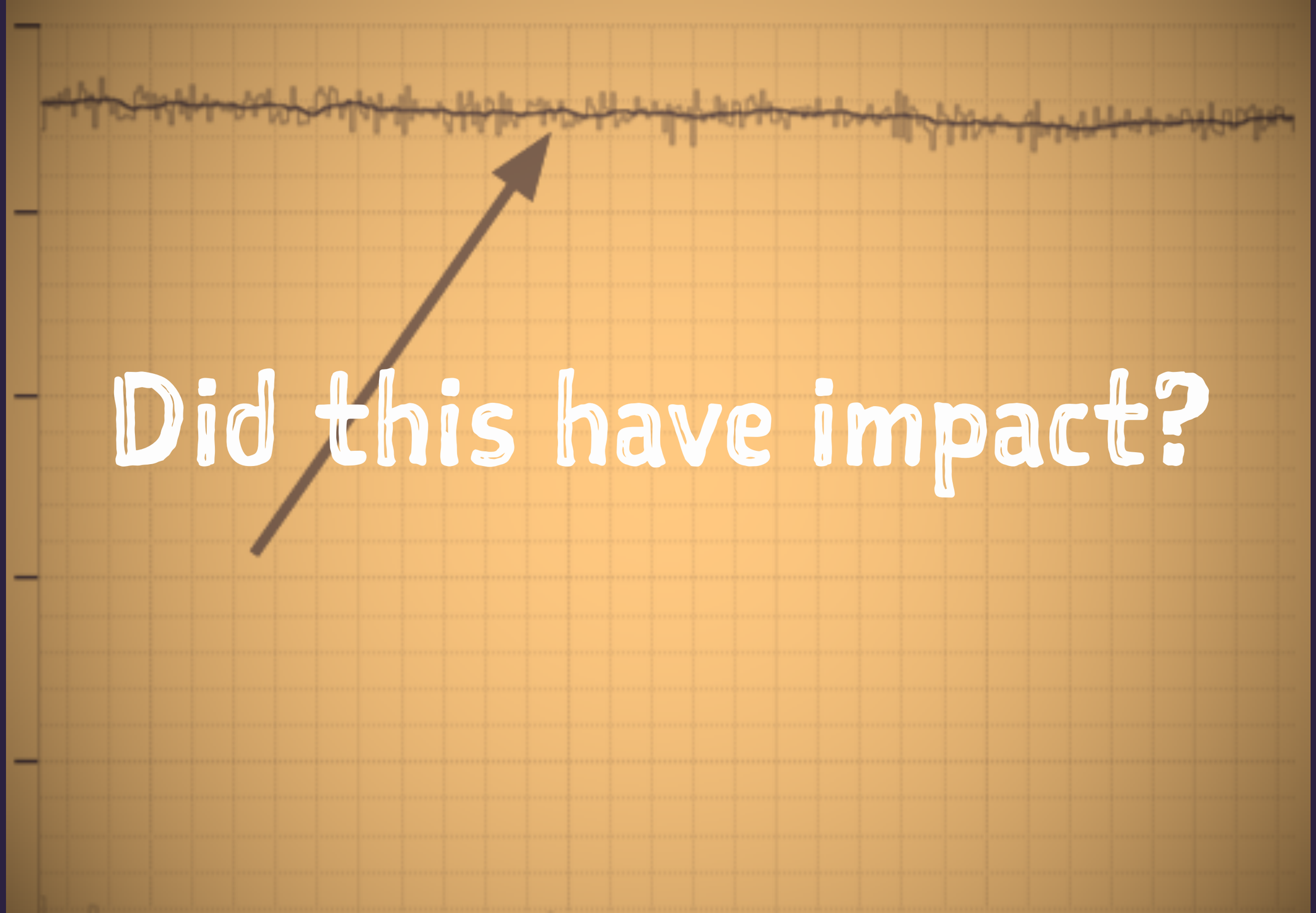
Example: Inject failure for 10% of customer traffic

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 II: Chaos as experimentation


ChAP

Chaos Automation Platform

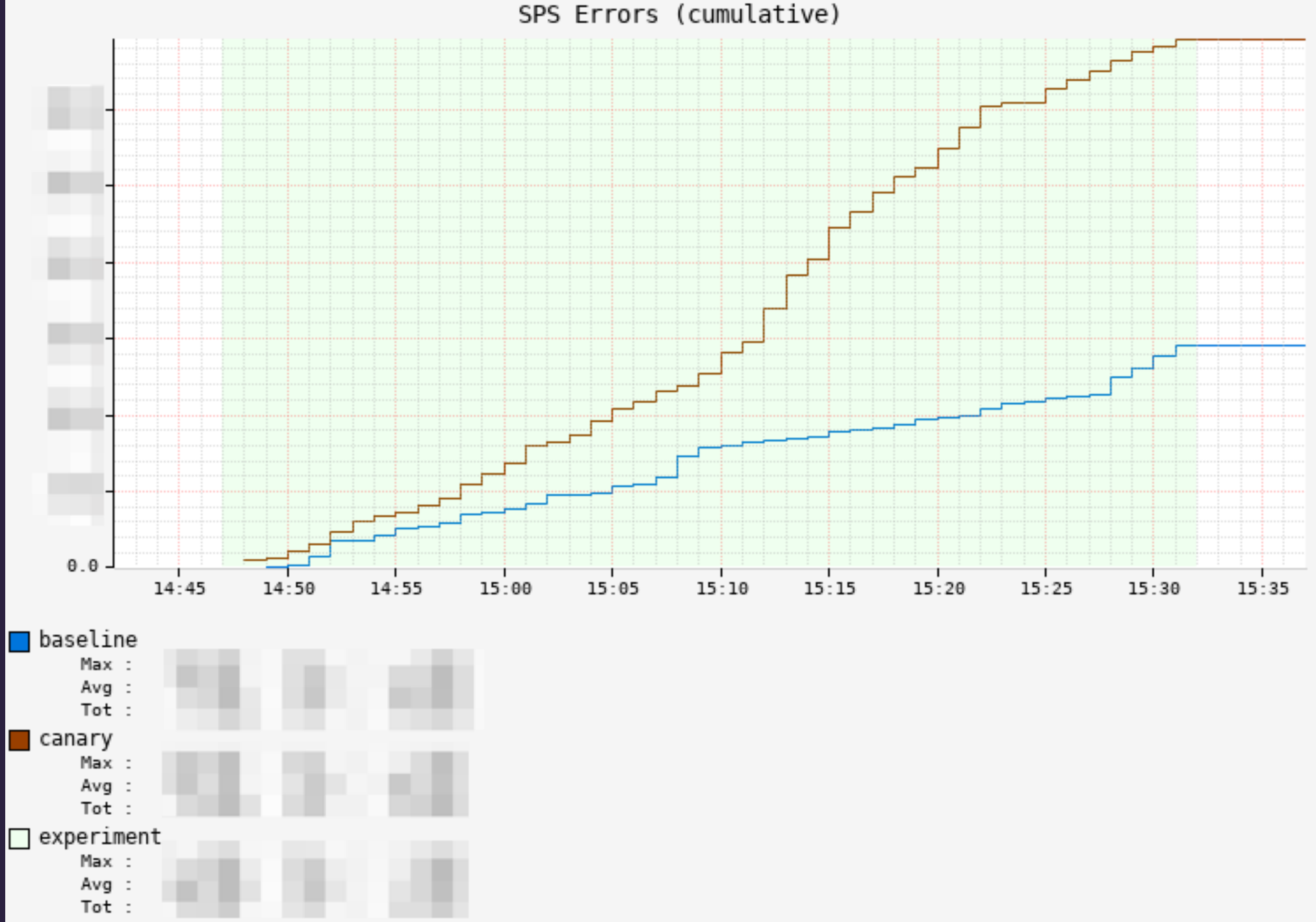
Want: clear signal if failure injection having negative impact...

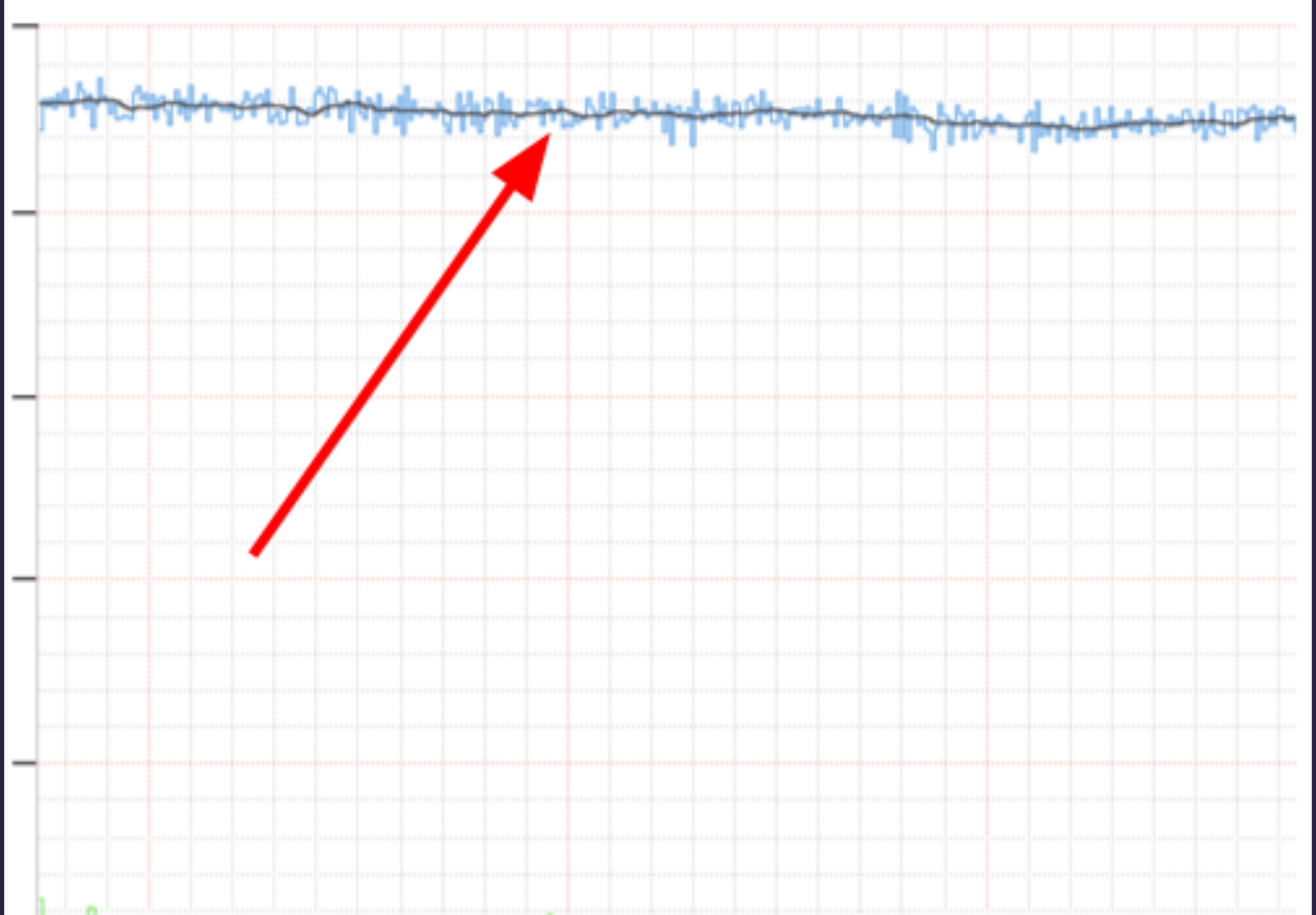
...on customers...

...and on services

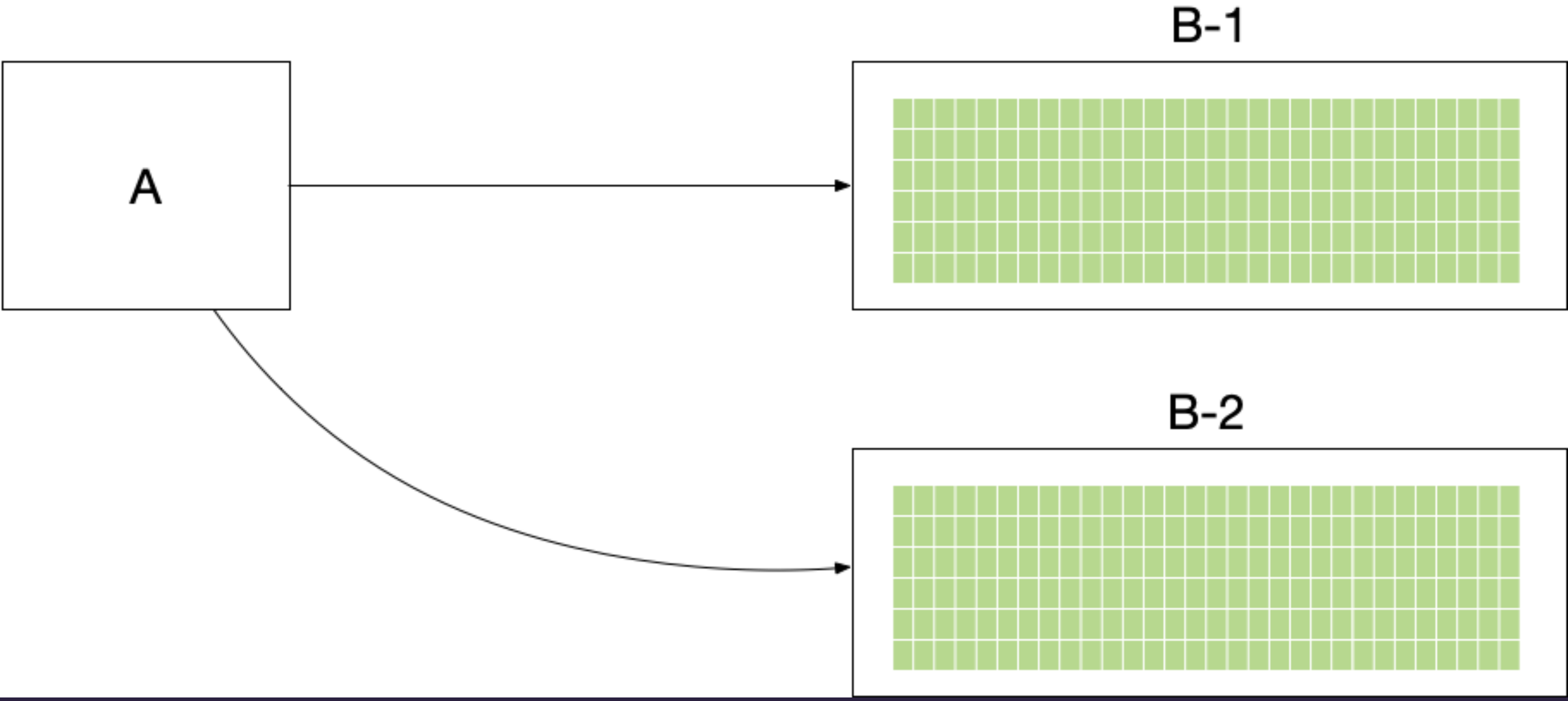
Big idea: stickiness

Failure injection sessions are sticky to users

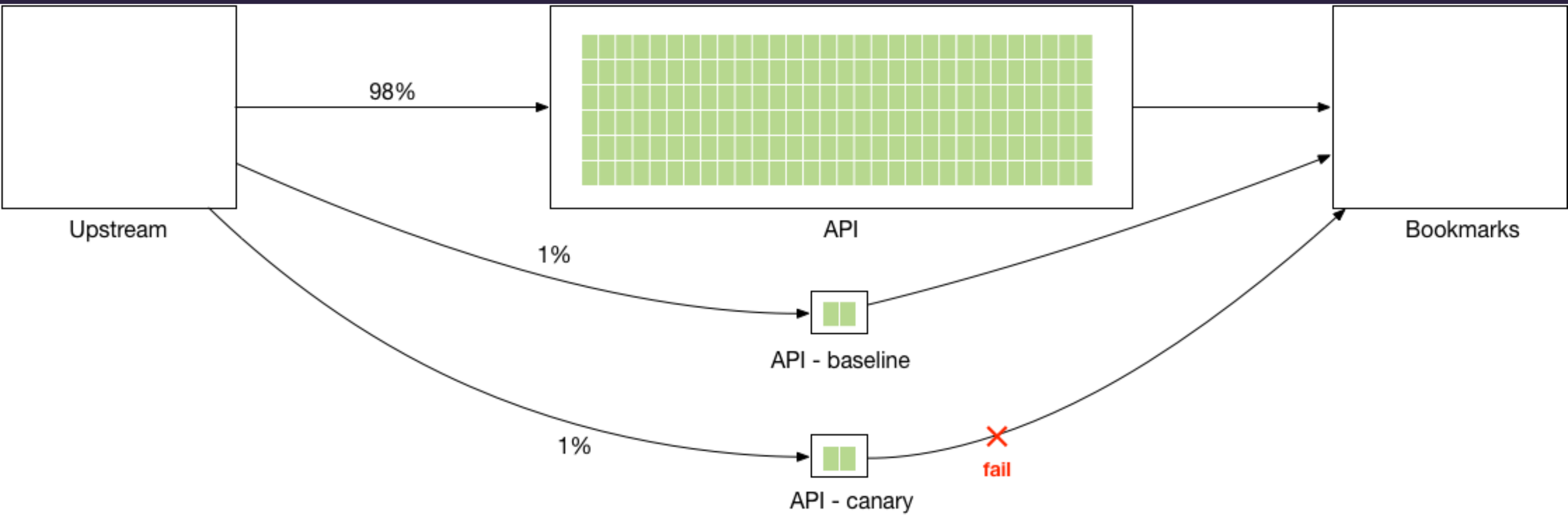

SPS Errors (cumulative)
0.0
14:45
14:50
14:55
15:00
15:05
15:10
15:15
15:20
15:25
15:30
15:35
baseline
Max :
Avg :
Tot :
canary
Max :
Avg :
Tot :
experiment
Max :
Avg :
Tot :

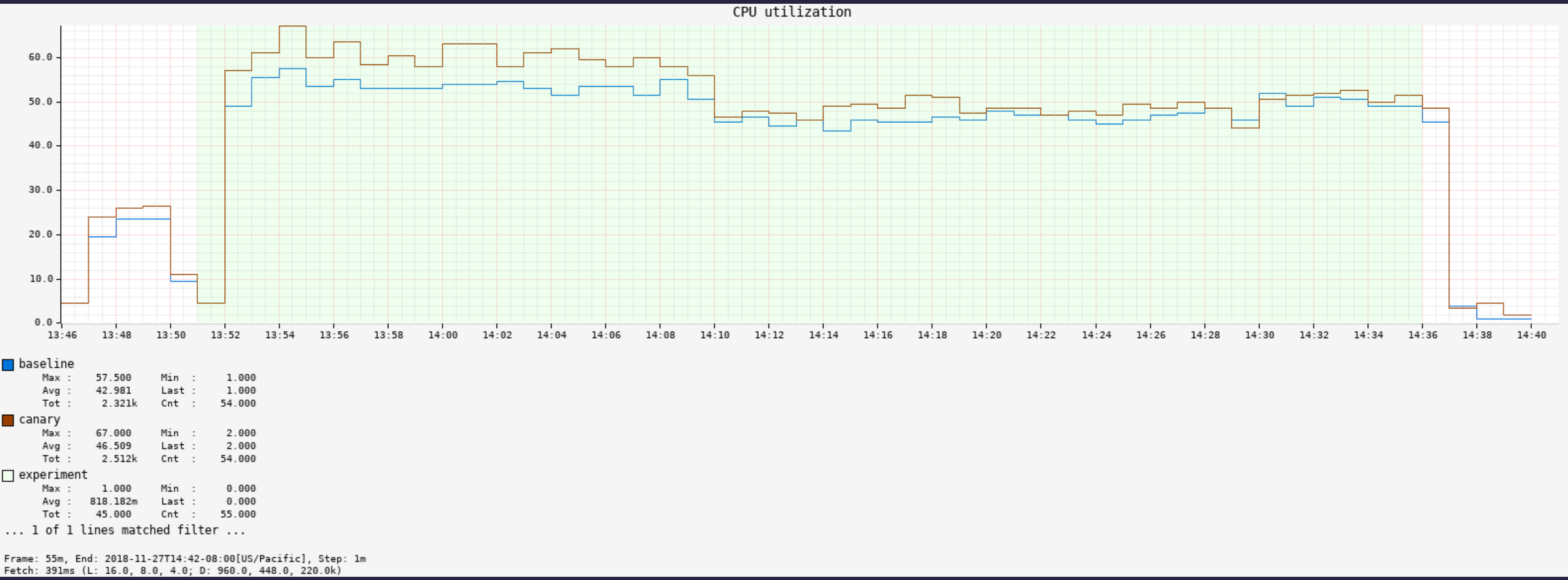


Failure injection sessions are sticky to clusters


B-1
A
B-2

We can do controlled experiments!


98%
Upstream
API
Bookmarks
1%
API - baseline
1%
fail
API - canary



baseline
Max : 57.500 Min : 1.000
Avg : 42.981 Last : 1.000
Tot : 2.321k Cnt : 54.000

canary
Max : 67.000 Min : 2.000
Avg : 46.509 Last : 2.000
Tot : 2.512k Cnt : 54.000

experiment
Max : 1.000 Min : 0.000
Avg : 818.182m Last : 0.000
Tot : 45.000 Cnt : 55.000

... 1 of 1 lines matched filter ...

Frame: 55m, End: 2018-11-27T14:42-08:00[US/Pacific], Step: 1m
Fetch: 391ms (L: 16.0, 8.0, 4.0; D: 960.0, 448.0, 220.0k)

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

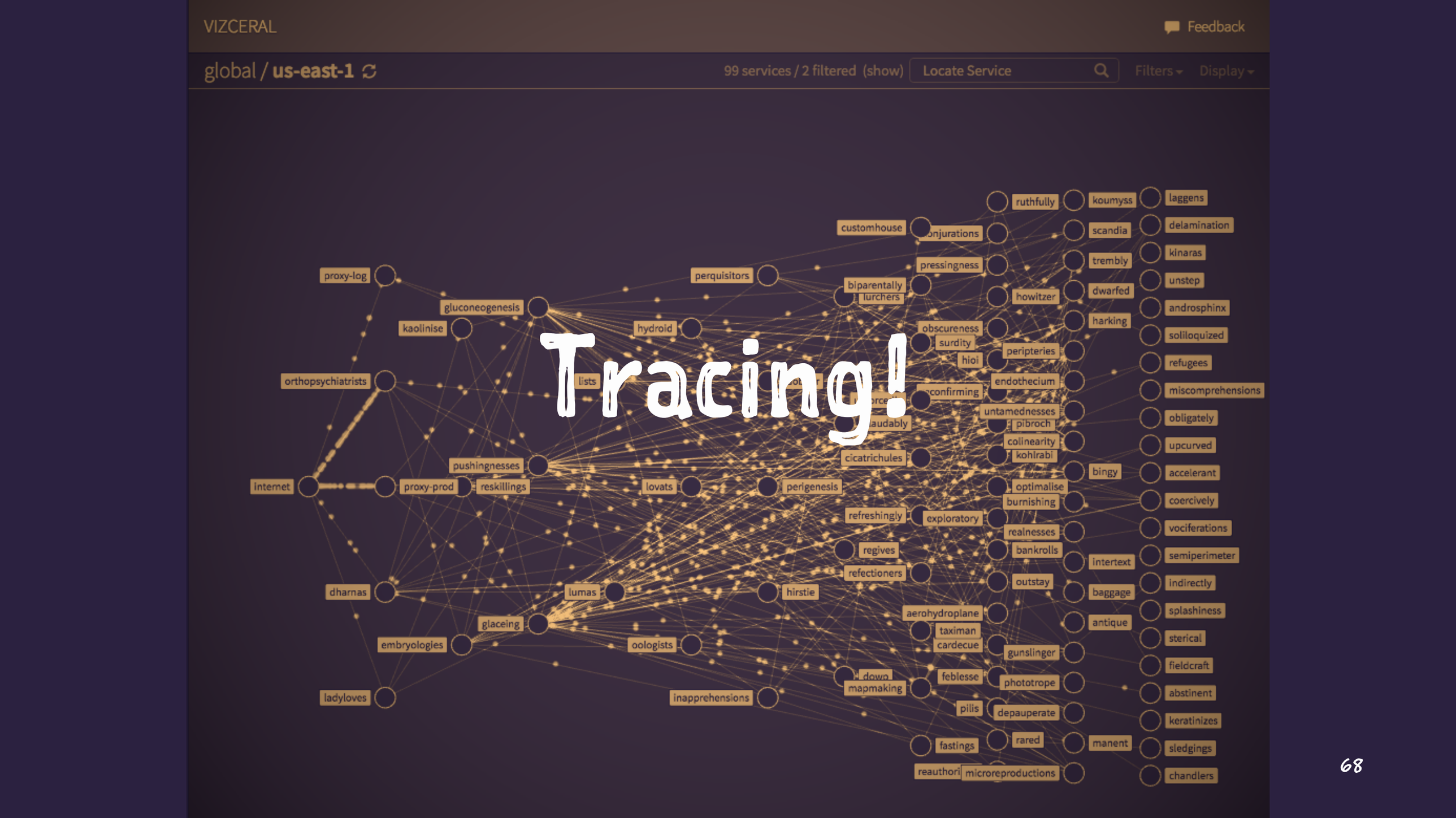
Actively engage with multiple teams

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?

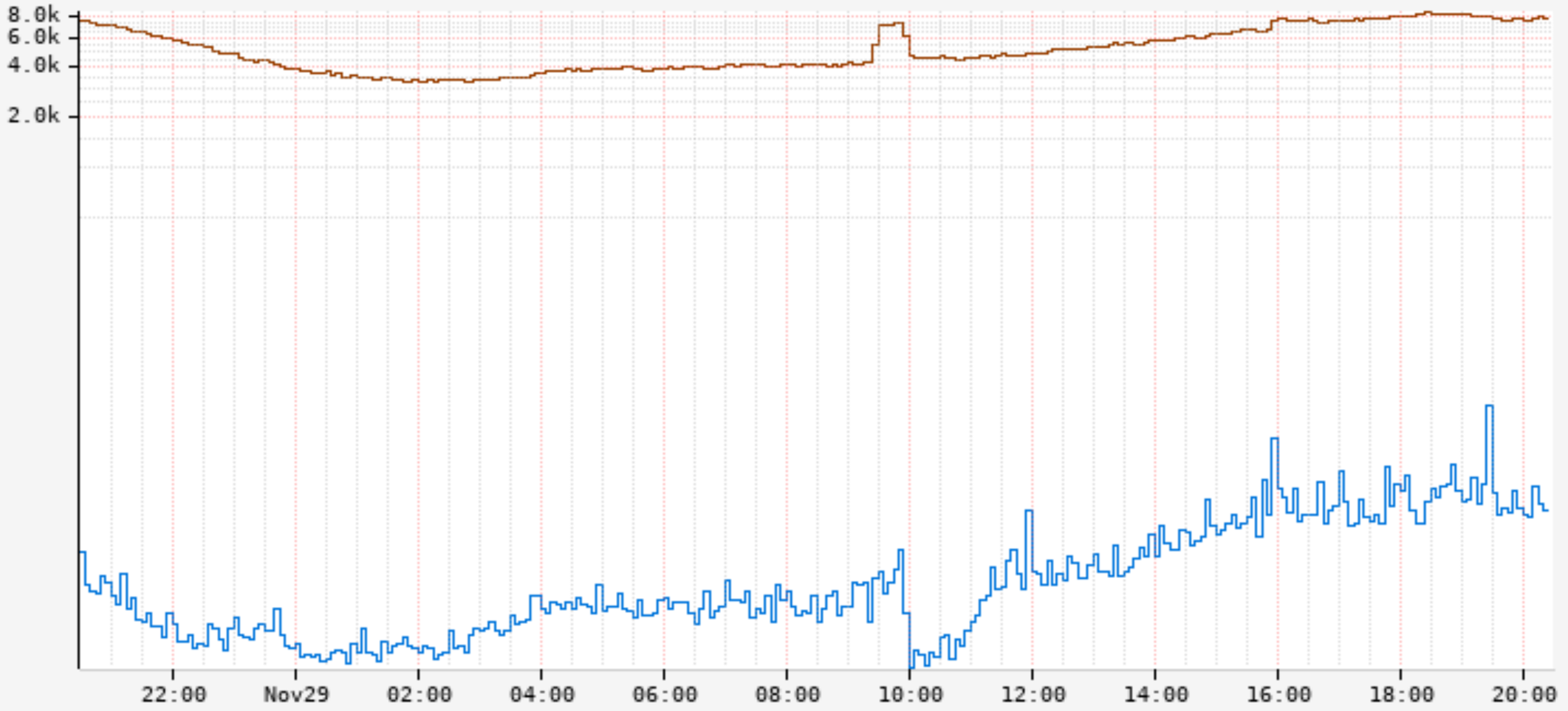

VIZCERAL
Feedback
global / us-east-1
99 services / 2 filtered (show)
Locate Service
Filters
Display
Tracing!
proxy-log
gluconeogenesis
kaolinise
orthopsychiatrists
internet
proxy-prod
reskillings
pushingnesses
dharnas
embryologies
ladyloves
glaceing
lumas
perquisitors
hydroid
lists
lovats
perigenesis
hirstie
oologists
inapprehensions
customhouse
biparentally
lurchers
pressingness
obscureness
surdity
hioi
cicatrichules
refreshingly
regives
refectioners
aerohydroplane
taximan
cardecue
feblesse
mapmaking
pilis
fastings
ruthfully
howitzer
peripteries
endothecium
untamednesses
pibroch
colinearity
kohlrabi
optimalise
burnishing
exploratory
realnesses
bankrolls
outstay
gunslinger
phototrope
depauperate
rared
microreproductions
koumyss
scandia
trembly
dwarfed
harking
bingy
intertext
baggage
antique
manent
laggens
delamination
kinaras
unstep
androsphinx
soliloquized
refugees
miscomprehensions
obligately
upcurved
accelerant
coercively
vociferations
semiperimeter
indirectly
splashiness
sterical
fieldcraft
abstinent
keratinizes
sledgings
chandlers

Which RPCs do we believe are safe to fail?

Heuristics!

Is there a fallback?

Does the fallback ever get invoked?



countSuccess

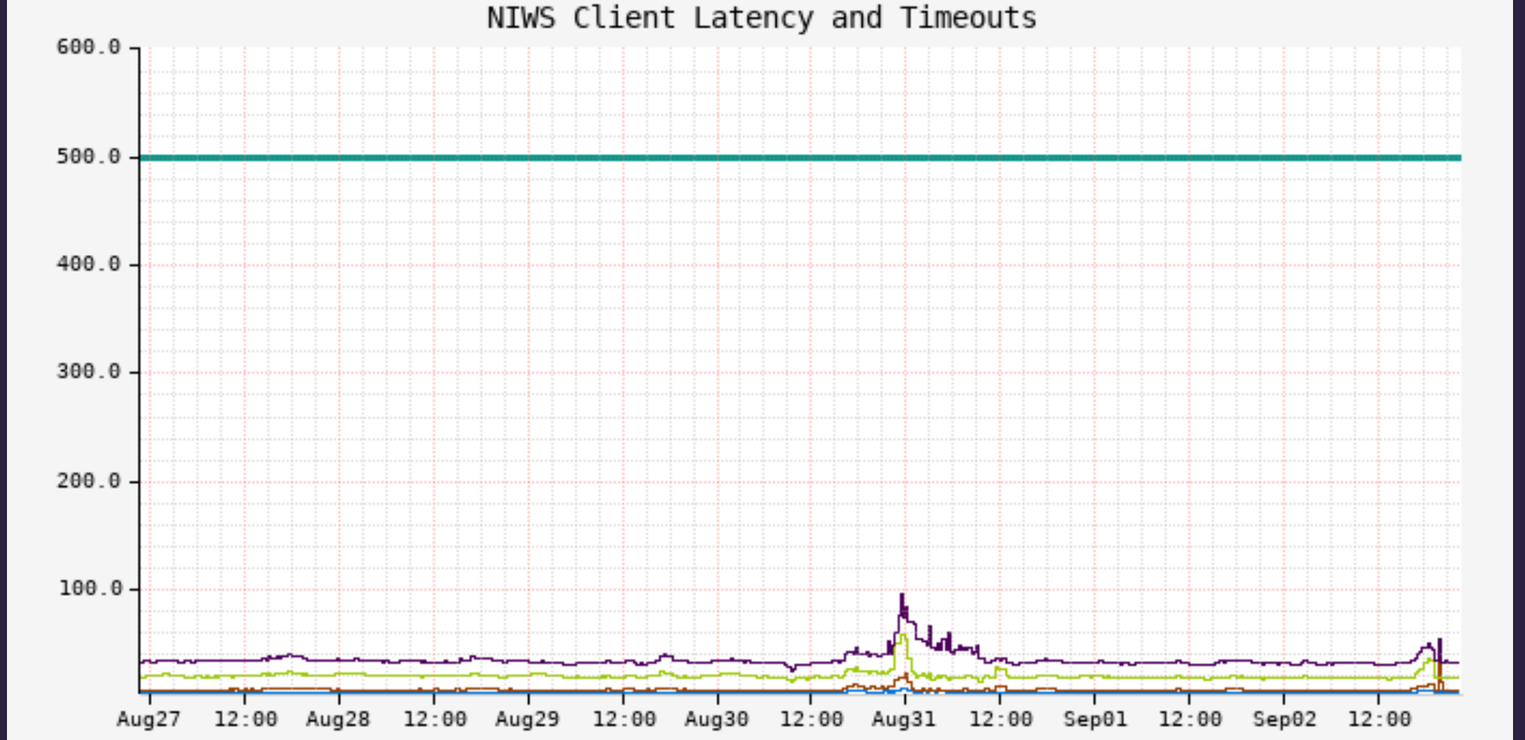
Max :	8.487k	Min :	3.258k
Avg :	5.321k	Last :	7.870k
Tot :	1.527M	Cnt :	287.000

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?



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

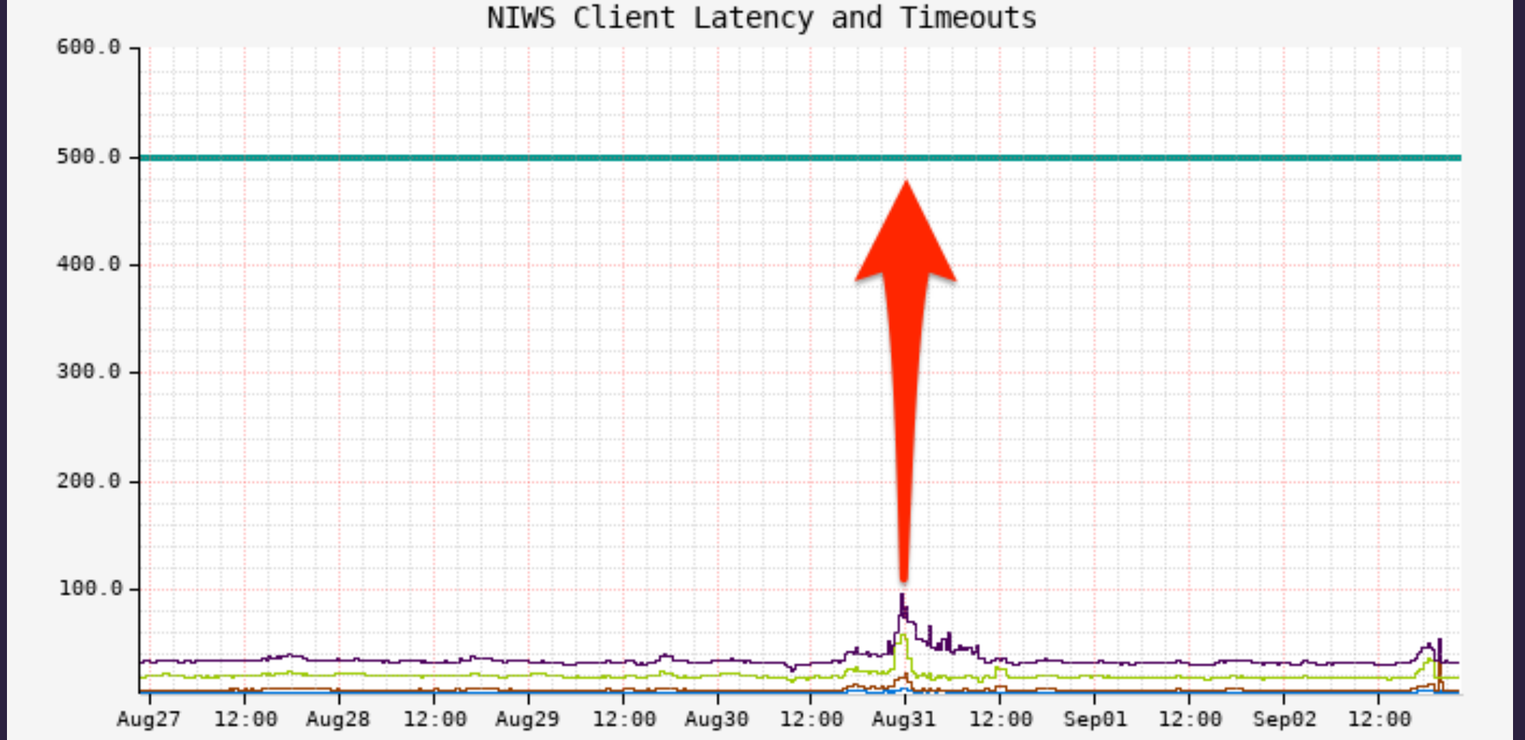
95th Percentile Latency
Max : 45.888 Min : 5.637
Avg : 7.806 Last : 7.527
Tot : 3.926k Cnt : 503.000

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)



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

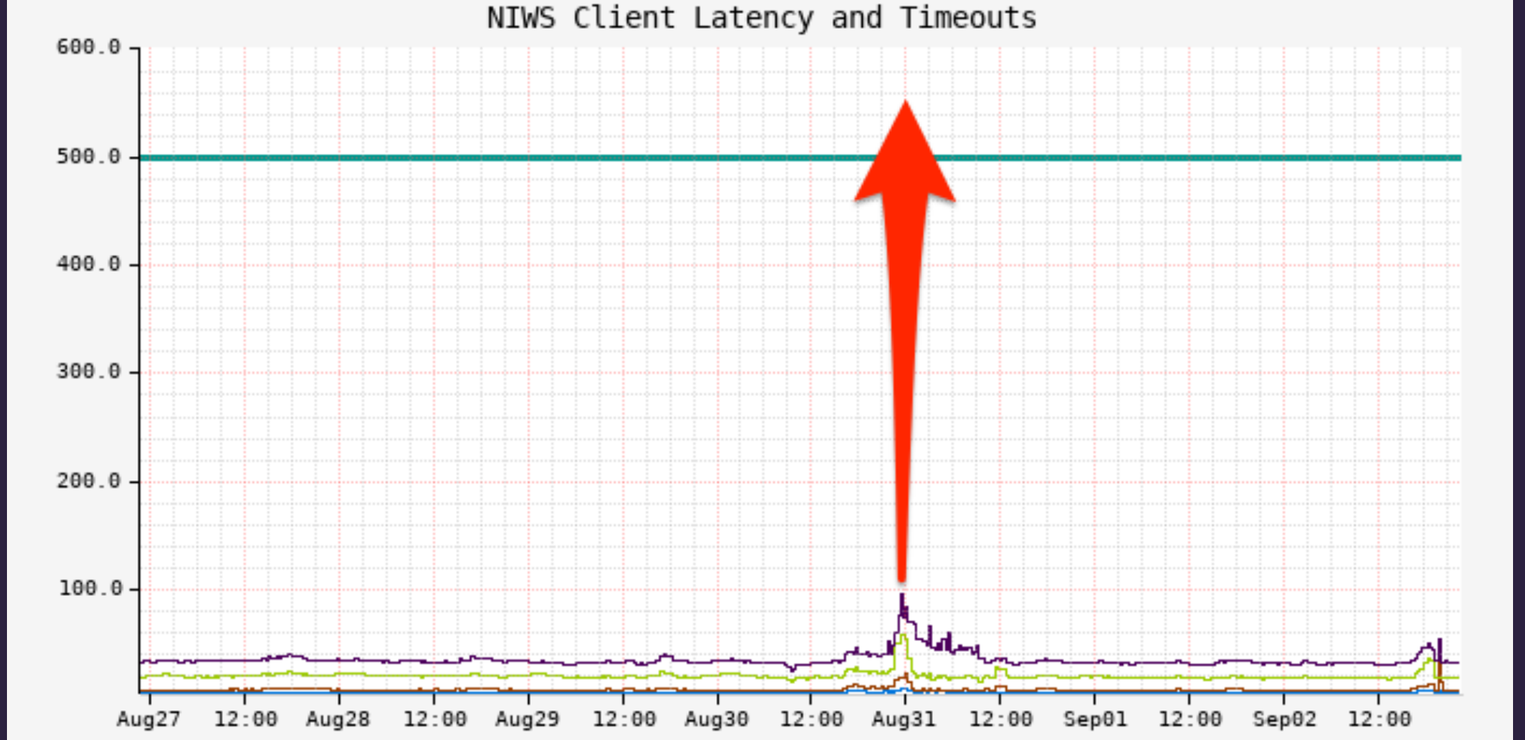
95th Percentile Latency
Max : 45.888 Min : 5.637
Avg : 7.806 Last : 7.527
Tot : 3.926k Cnt : 503.000

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)



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 III: Lessons learned

Safety

It needs to be safe, or nobody will use it

Safe = limited impact

https://en.wikipedia.org/wiki/Edsger_W._Dijkstra#/media/File:Edsger_Wybe_Dijkstra.jpg


No!

Safety adds complexity

$WithinLimit \triangleq Sum(running) \leq TrafficLimit$

$$
\begin{aligned}
TypeOK \triangleq\ & \land queue \in \text{SUBSET } Runs \\
& \land owned \in \text{SUBSET } Runs \\
& \land running \in \text{SUBSET } Runs \\
& \land traffic \in [Runs \to Nat \setminus \{0\}] \\
& \land candidate \in [ProcSet \to Runs \cup \{NoRun\}] \\
& \land known \in [ProcSet \to \text{SUBSET } Runs] \\
& \land pc \in [ProcSet \to \{\text{“p1”}, \text{“p2”}, \text{“p3”}, \text{“Done”}\}]
\end{aligned}
$$

$$
\begin{aligned}
Inv \triangleq\ & \land TypeOK \\
& \land \forall i \in ProcSet : known[i] \subseteq owned \\
& \land \forall i \in ProcSet : \lor candidate[i] = NoRun \\
& \qquad\qquad\qquad \lor candidate[i] \in owned \\
& \land \forall run \in running : \exists i \in ProcSet : \land pc[i] = \text{“Done”} \\
& \qquad\qquad\qquad\qquad\qquad\qquad \land candidate[i] = run \\
& \land \forall i, j \in ProcSet : \lor known[i] \subseteq known[j] \\
& \qquad\qquad\qquad\qquad \lor known[j] \subseteq known[i] \\
& \land WithinLimit
\end{aligned}
$$

ASSUME $NumWorkersInNat \triangleq NumWorkers \in Nat \setminus \{0\}$
ASSUME $TrafficLimitInNat \triangleq TrafficLimit \in Nat \setminus \{0\}$

LEMMA $SumPrime \triangleq \forall S \in \text{SUBSET } Runs : (Sum(S))' = Sum(S')$

LEMMA $EmptySumIsZero \triangleq Sum(\{\}) = 0$

LEMMA $SumInNat \triangleq \forall S \in \text{SUBSET } Runs : Sum(S) \in Nat$

THEOREM $Spec \Rightarrow \Box WithinLimit$
⟨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. $\forall i \in ProcSet : known[i] \subseteq owned$
    OBVIOUS
  ⟨2⟩3. $\forall i \in ProcSet : \lor candidate[i] = NoRun$
    $\lor candidate[i] \in owned$

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)
```

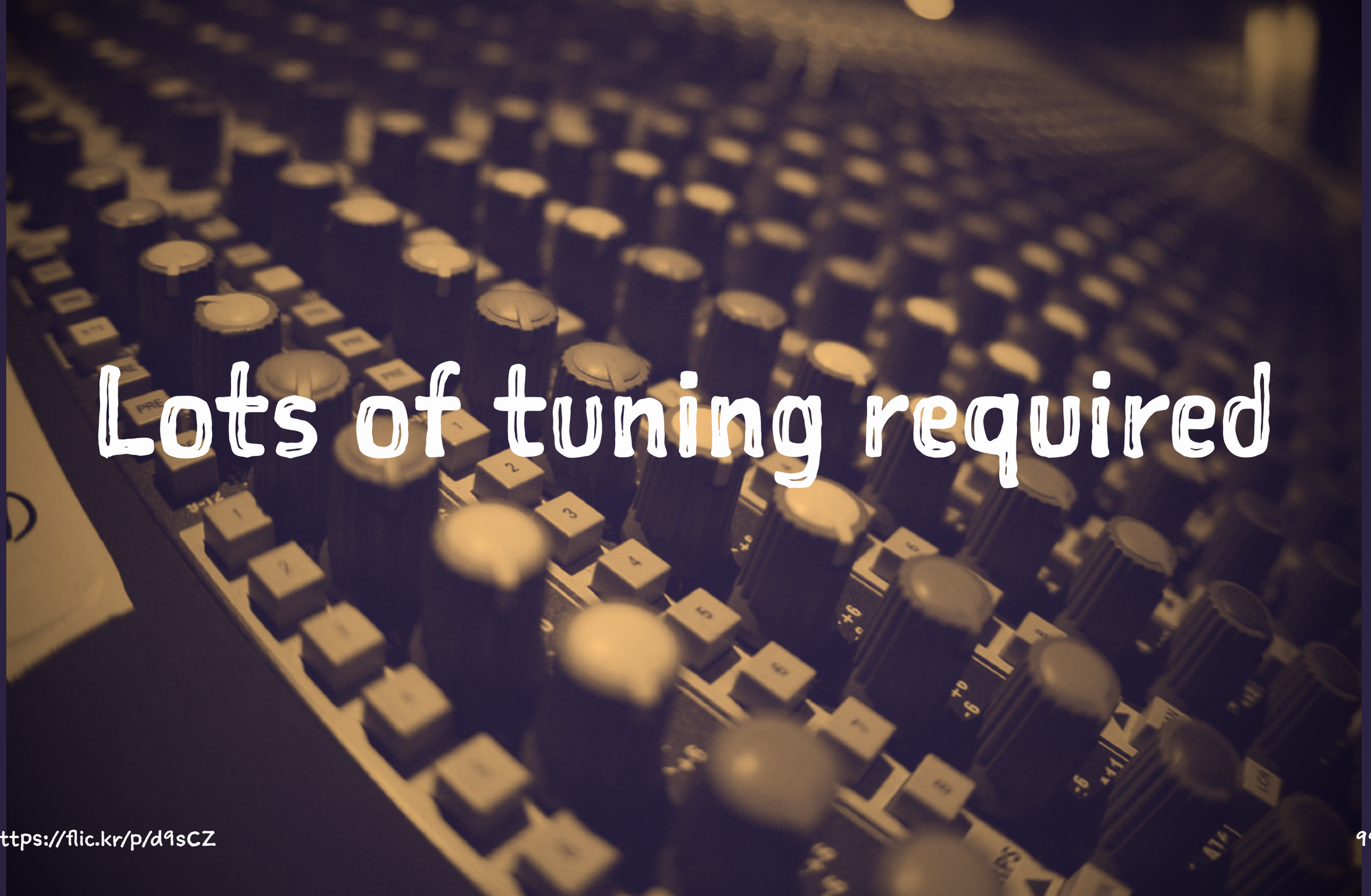
ChAP isn't a "black box"

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



https://flic.kr/p/d9sCZ

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

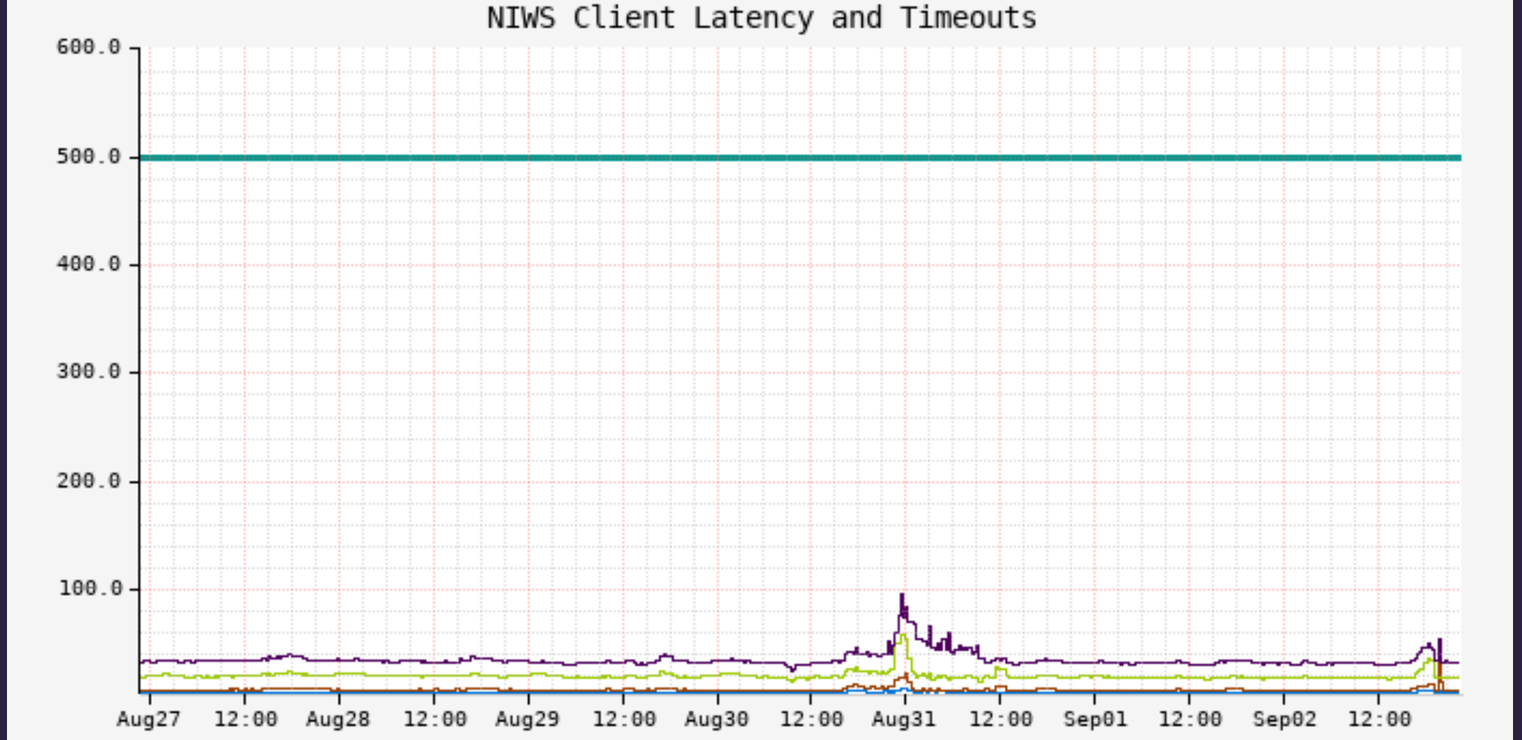
Java -> Node.js

REST -> gRPC

VMs -> containers

Unexpected benefits

Info for experiment generation was useful to service owners



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)

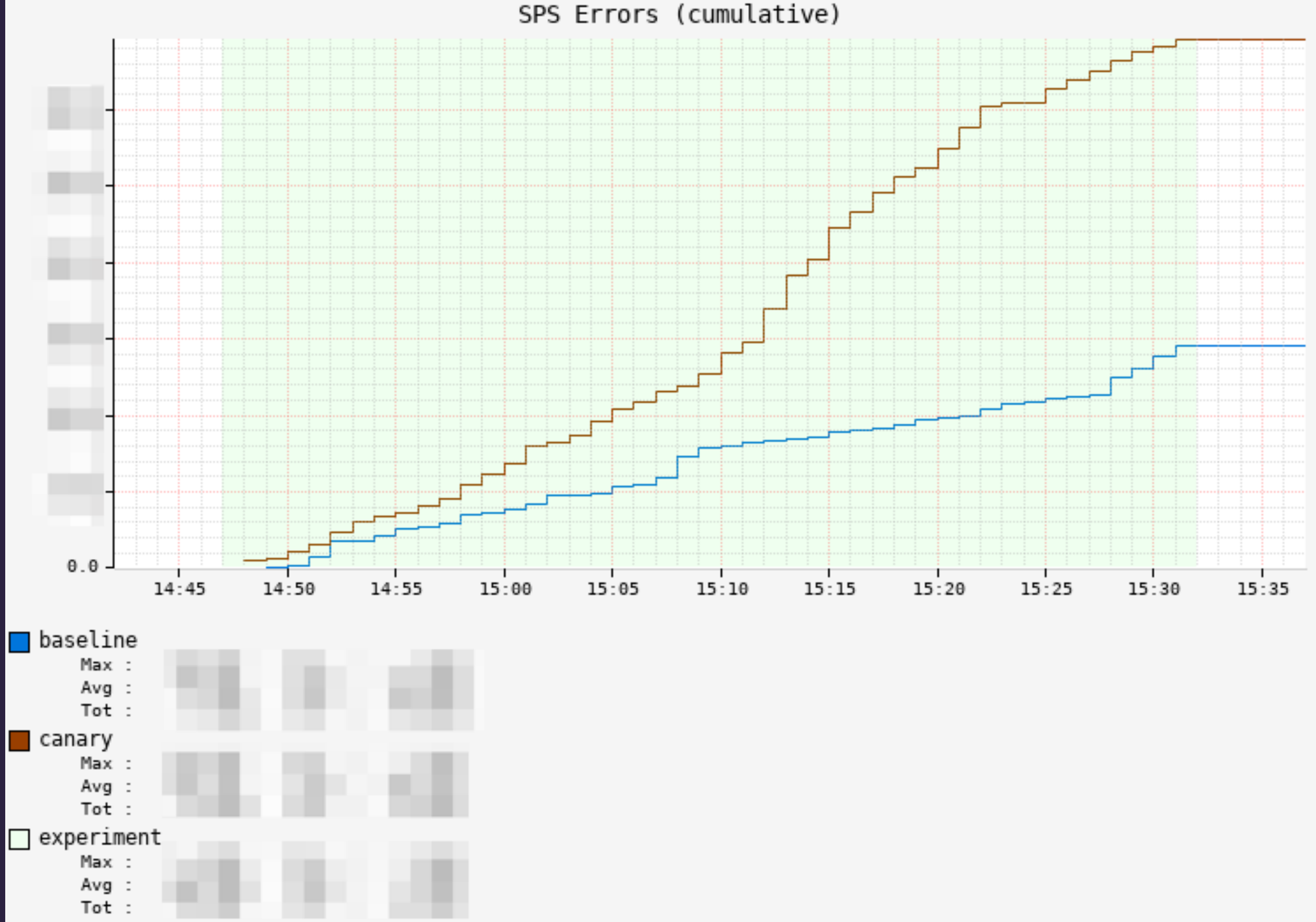

SPS Errors (cumulative)
0.0
14:45
14:50
14:55
15:00
15:05
15:10
15:15
15:20
15:25
15:30
15:35
baseline
Max :
Avg :
Tot :
canary
Max :
Avg :
Tot :
experiment
Max :
Avg :
Tot :

Image credits

- "Knobs", Ian Harding, CC-BY-NC-SA 2.0: https://flic.kr/p/d9sCZ
- "Portrait of Edsger W. Dijkstra", Hamilton Richards, CC BY-SA 3.0: https://en.wikipedia.org/wiki/Edsger_W._Dijkstra#/media/File:EdsgerWybeDijkstra.jpg