Antics, drift and chaos

Lorin Hochstein
Chaos Team, Netflix
@lhochstein

Executes tests to warm

One day...

Add a new test

@Category(FunctionalTest.class)

```
Exception in thread "foo"
java.lang.ClassNotFoundException
```

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.

Result: execution of unit test led to an outage

Moral: use unit tests sparingly, for they are dangerous

~~Moral: use unit tests sparingly,~~
~~for they are dangerous~~

Complex systems exhibit unexpected behavior

Failure

System failure

Outages

Incidents

How

@lhochstein

Why

@lhochstein

What

@lhochstein

Act I: Antics

Complex systems exhibit unexpected behavior

— John Gall

Generalized Uncertainty Principle

Systems display antics

— John Gall

1. Error handling

Any large system is going to be operating most of the time in failure mode

— John Gall

Ding Yuan et al., Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems, OSDI 2014.

Almost all catastrophic failures (92%) are the result of incorrect handling of non-fatal errors explicitly signaled in software

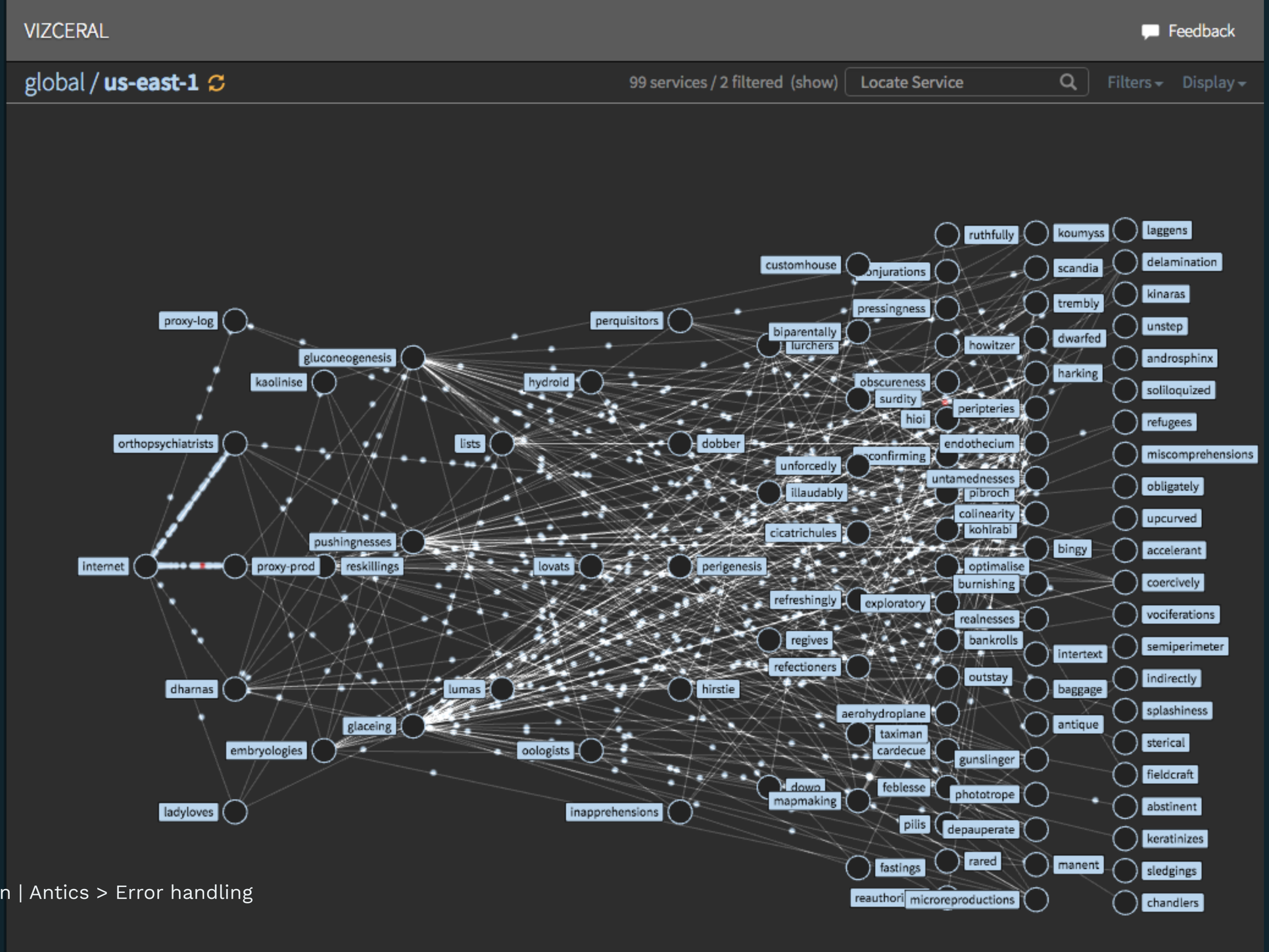
Problems are not the problem;
coping is the problem.

— John Gall

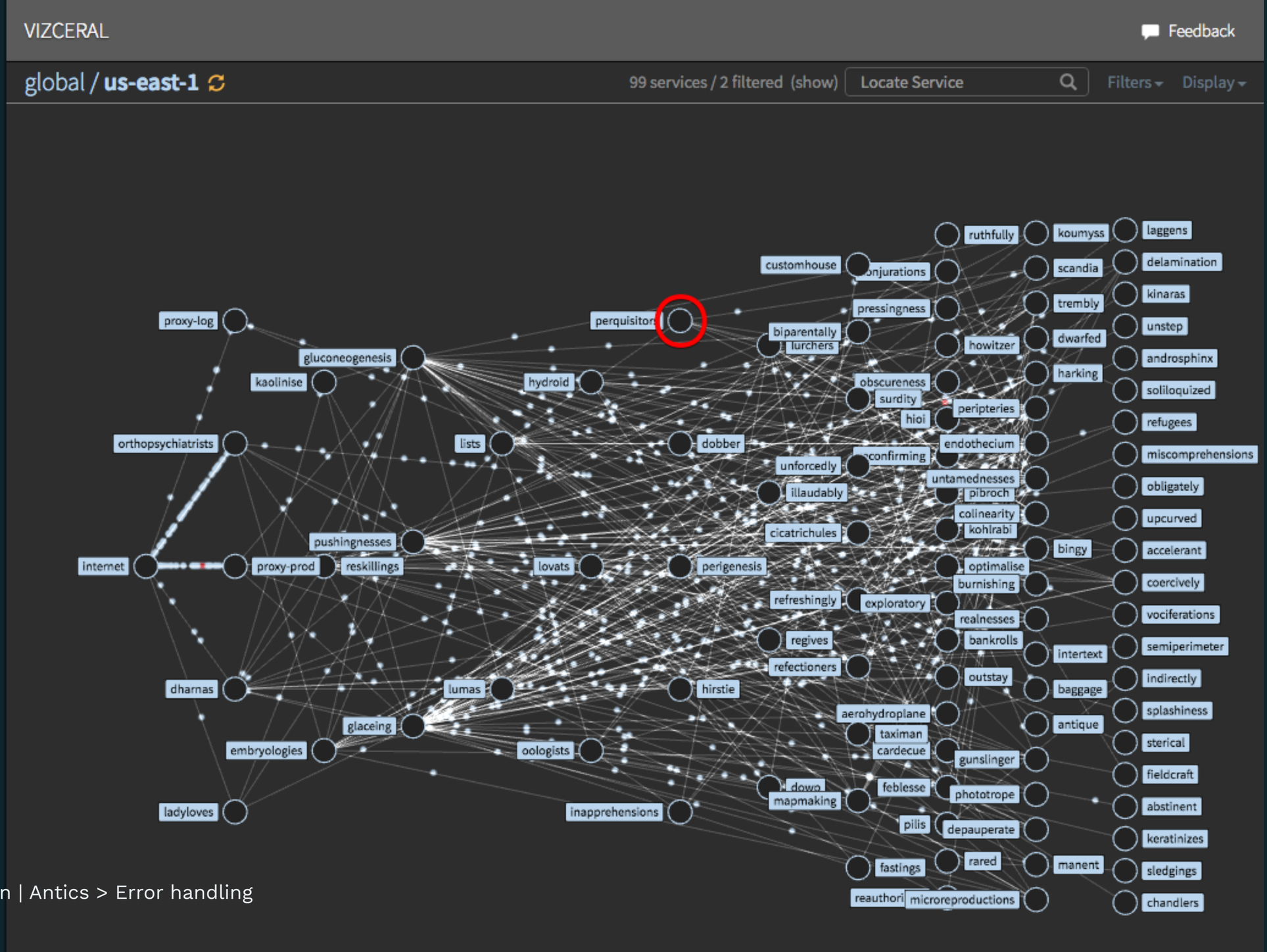
```
C:\> DIR A:

Not ready reading Drive A
Abort, Retry, Fail?
```

Scenario



@lhochstein | Antics > Error handling



@lhochstein | Antics > Error handling

Service becomes latent

Clients timeout

Clients retry

Load increases

Latency increases

More clients retry

Ryan Huang et al., Gray failure: the Achilles' heel of cloud-scale systems, HotOS 2017

Zhenyu Guo et al., Failure Recovery: When the Cure Is Worse Than the Disease, HotOS 2013

2. Support systems

Why does Netflix need so many engineers?

Why does Netflix need so many engineers?

We're hiring! Come visit our booth!

Operational fallacy: the system itself does not do what it says

— John Gall

Harvard is really a $40 billion tax-free hedge fund with a very large marketing and PR arm called Harvard University.

— Jim Manzi

Netflix is a monitoring company, that as an interesting an unexpected byproduct also streams movies

— Adrian Cockroft (attributed)

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.

Non-critical service failed

Log messages increased

Log messages sent to Kafka

Lock shared by app threads

Lock contention

Logging took down prod

3. Mitigation

AWS S3 Outage of Feb 2017

S3 billing process was slow

AWS eng tried to remove some servers

Command input entered incorrectly

Lorin's conjecture

Most major incidents will be due to

1. Unexpected behavior of a support system
2. Attempt to mitigate a non-critical incident

Recap: Antics

Mechanisms that improve availability (error-handling, support systems, mitigation) **also create outages**

Act II: Drift

Broken parts and sloppy devs

"Be more careful"

Our technologies have got ahead of our theories

— Sidney Dekker

Drift into failure

1. Unruly technology

Software is hard to reason about

We can't model our systems

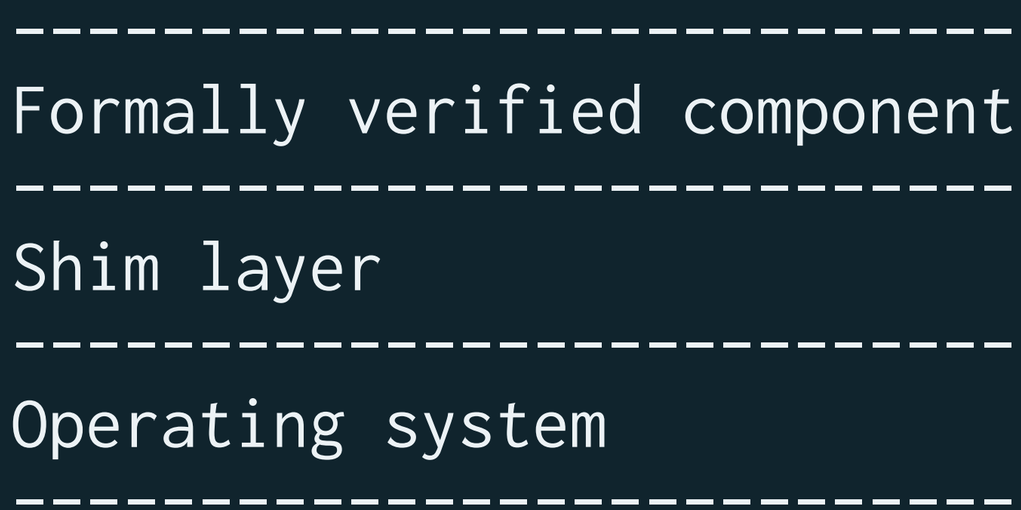
Fault-tolerance isn't composable.

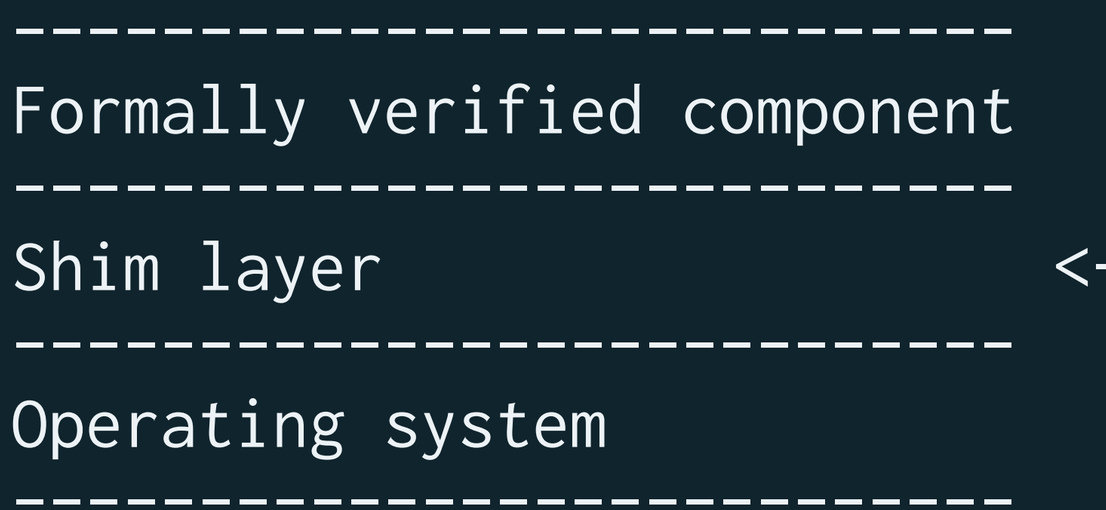
— Peter Alvaro

The mode of failure of a complex system cannot ordinarily be determined from its structure

— John Gall

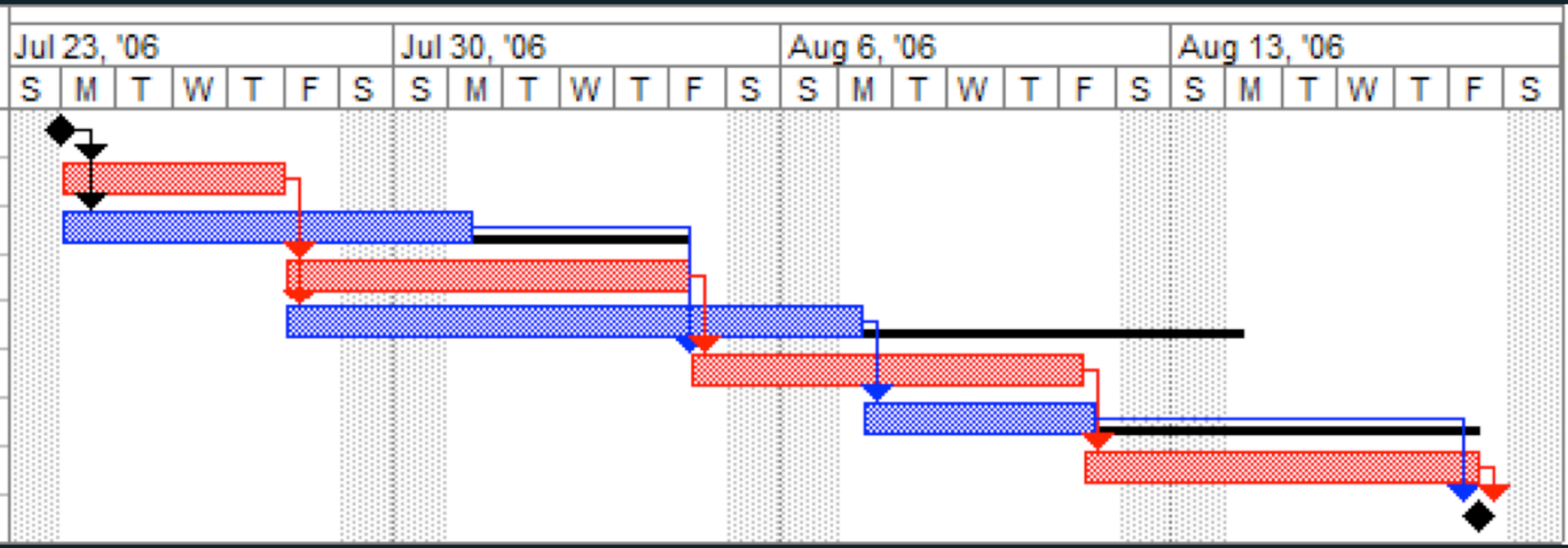
Pedro Fonseca et al., An Empirical Study on the Correctness of Formally Verified Distributed Systems, EuroSys 2017.


Formally verified component
Shim layer
Operating system


Formally verified component
Shim layer
<---- Most bugs are here
Operating system

2. Scarcity and competition

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



Efficiency vs thoroughness

ETTO Principle

— Erik Hollnagel

A temporary patch will very likely be permanent

— John Gall

3. Decrementalism

Drift happens in small steps

When is it OK to push to prod?

Failed canary probably OK if actual code change looks harmless

0operator

@sadoperator

Following

me: this change is pretty small, should be fine to deploy to production

narrator: but the change was not fine to deploy to production

12:40 PM - 25 Sep 2017

Normalization of deviance

— Diane Vaughan

4. Sensitive dependence on initial conditions

A complex system that works is inavariably found to have evolved from a simple system that worked

— John Gall

We make local decisions that have non-local impact


EVCache


Playback
URLs
EVcache


Playback
-X->
URLs
--->
EVcache
If URLs fails,
Playback has a fallback

One day...

Traffic spike


Playback
URLs
EVcache
NIC saturates

— EVcache client in URLs treats timeout as cache miss

— EVcache client in URLs treats timeout as cache miss

— Playback can't handle missing data scenario

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.

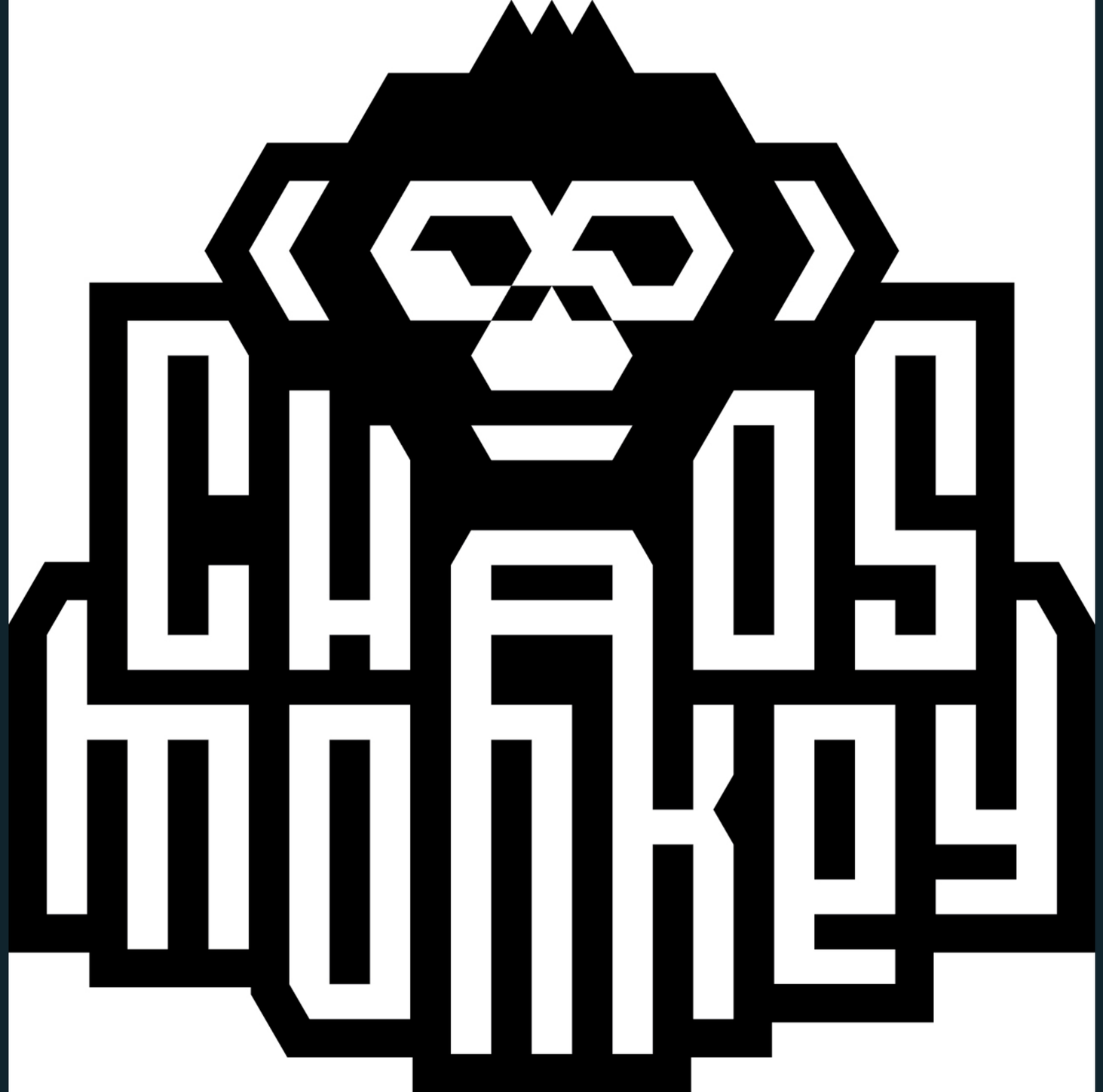
EVcache client defaults to treating errors as cache misses

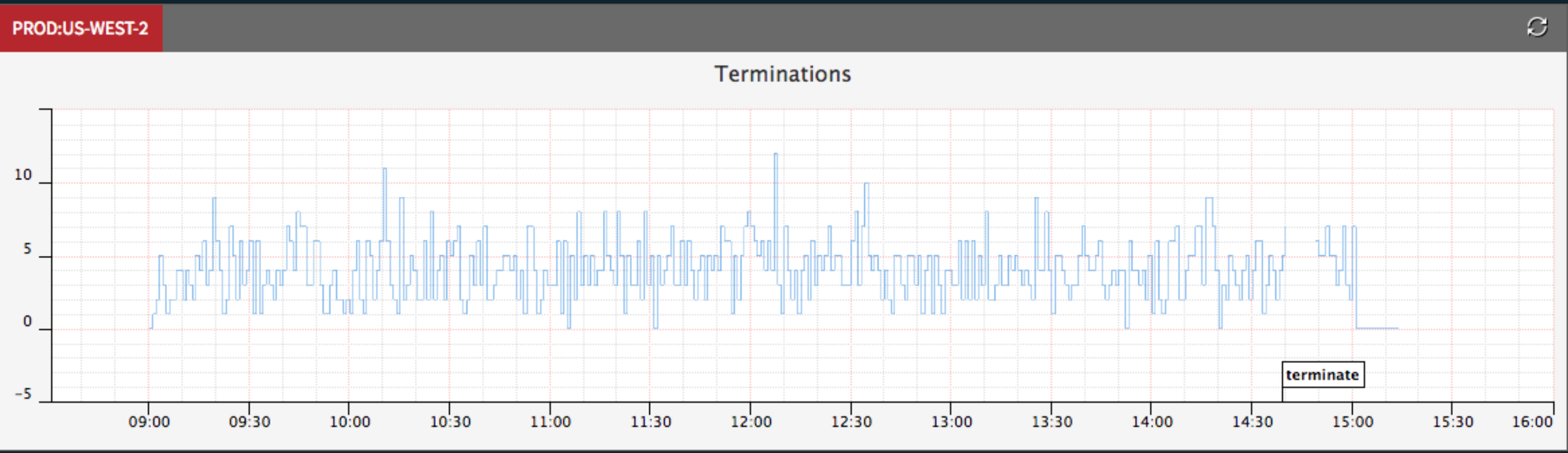
That's the correct behavior in most usages

Recap: Drift

The nature of software, how people behave under resource constraints, how people make local decisions, and history all contribute to system failure

Act III: Chaos


CHAOS
MONKEY


PROD:US-WEST-2
Terminations
10
5
0
-5
terminate
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

Make the wrong thing harder

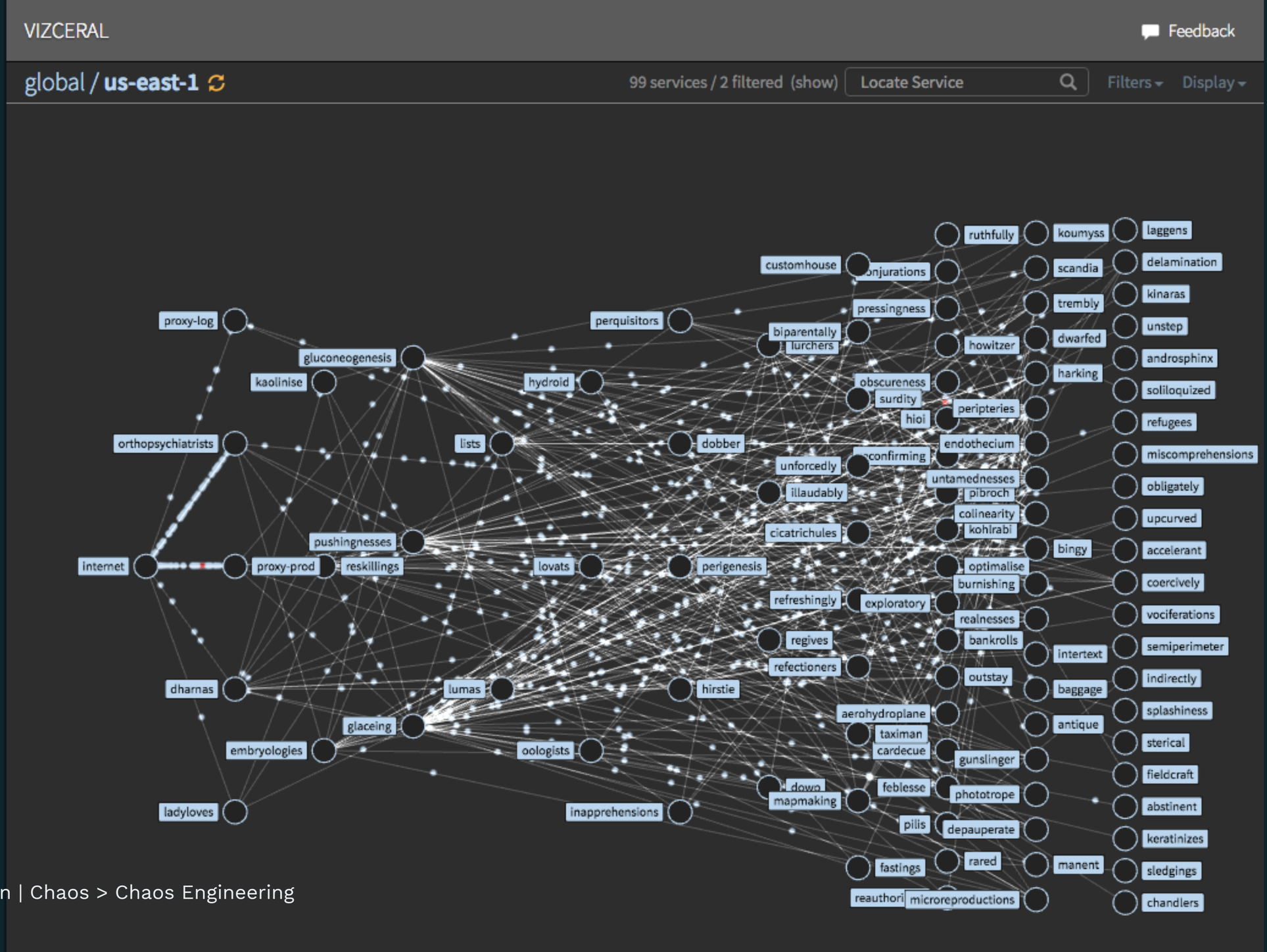
Chaos engineering

Find vulnerabilities before they become outages

In production

External validity

Risk: vulnerable to failure of non-critical services



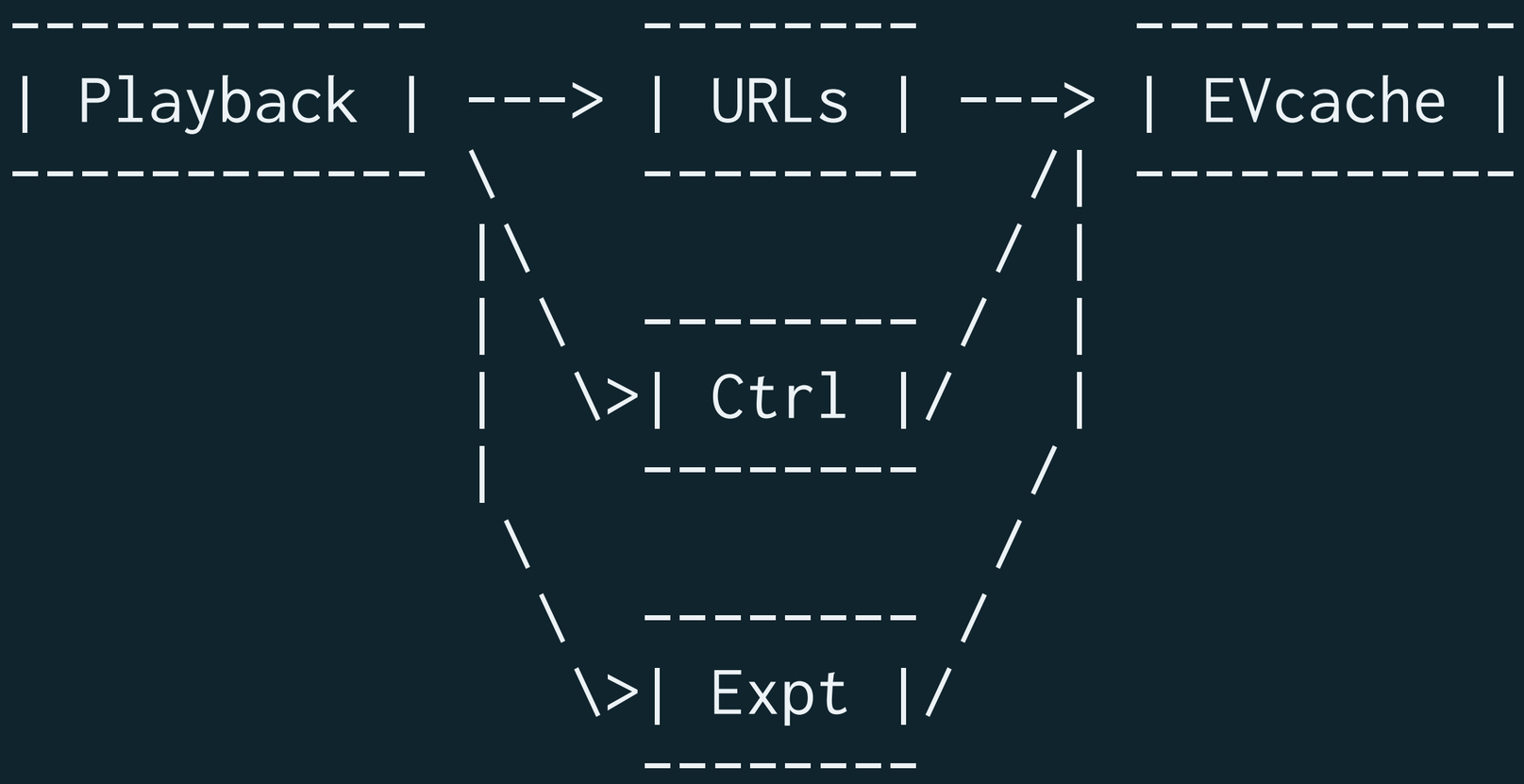
@lhochstein | Chaos > Chaos Engineering


ChAP


Playback
URLs
EVcache

Are we vulnerable to EVcache timeouts?

1. Clone URLs cluster to make two smaller clusters



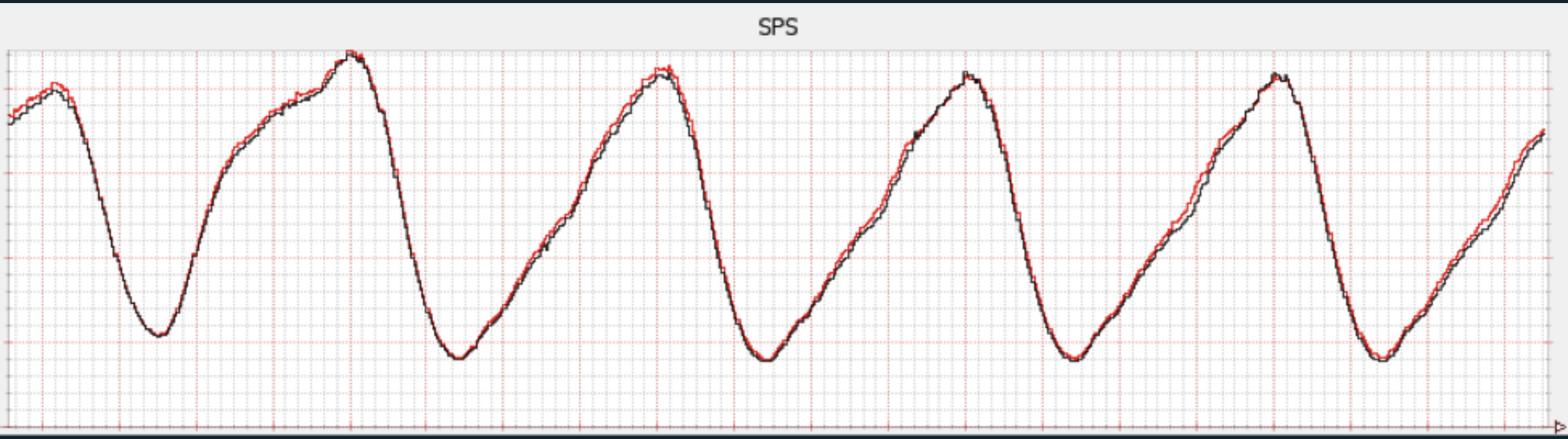
2. Route fraction of prod traffic to control and experiment clusters

3. Inject latency in calls from experiment cluster to EVcache

4. Measure differences between control & experiment clusters

principlesofchaos.org

1. Build a hypothesis around steady state behavior


SPS

2. Vary real-world events

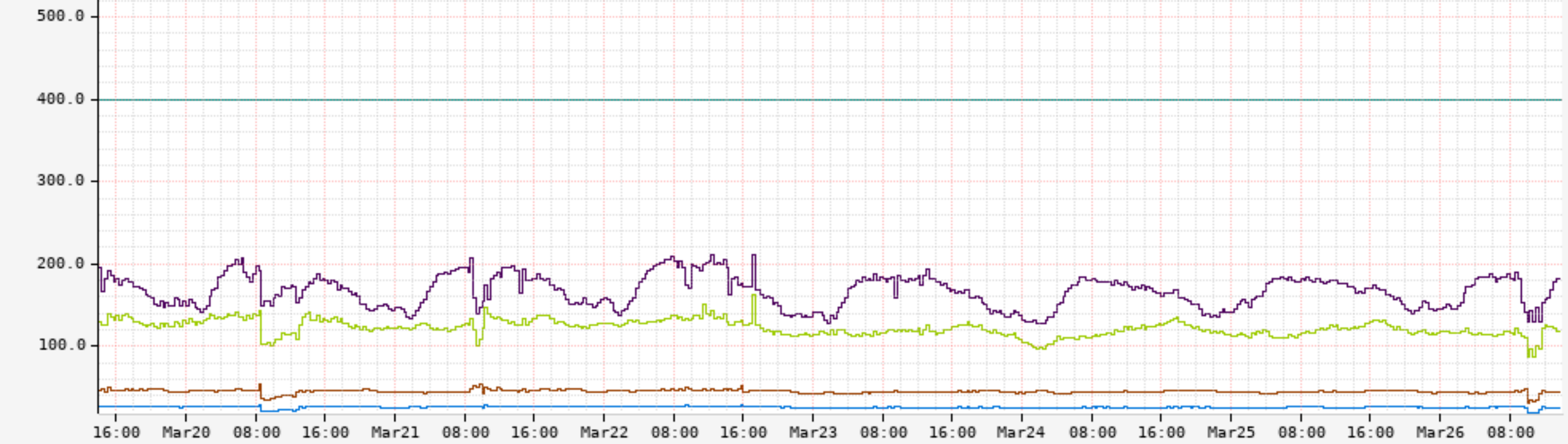
Fail RPC calls

Add latency to RPC calls

3. Run experiments in production

Route prod traffic to ChAP clusters

4. Automate experiments to run continuously



Average Latency

95th Percentile Latency

99th Percentile Latency

99.5th Percentile Latency

Timeout: 400 ms

5. Minimize blast radius

Route a small fraction of traffic

Stop early if impact detected

Takeaways

1. Systems behave pathologically

Chaos experiments can find pathologies

2. Reasonable human decisions can lead to dangerous states

Chaos provides incentives

3. Read these books