Evolving systems design

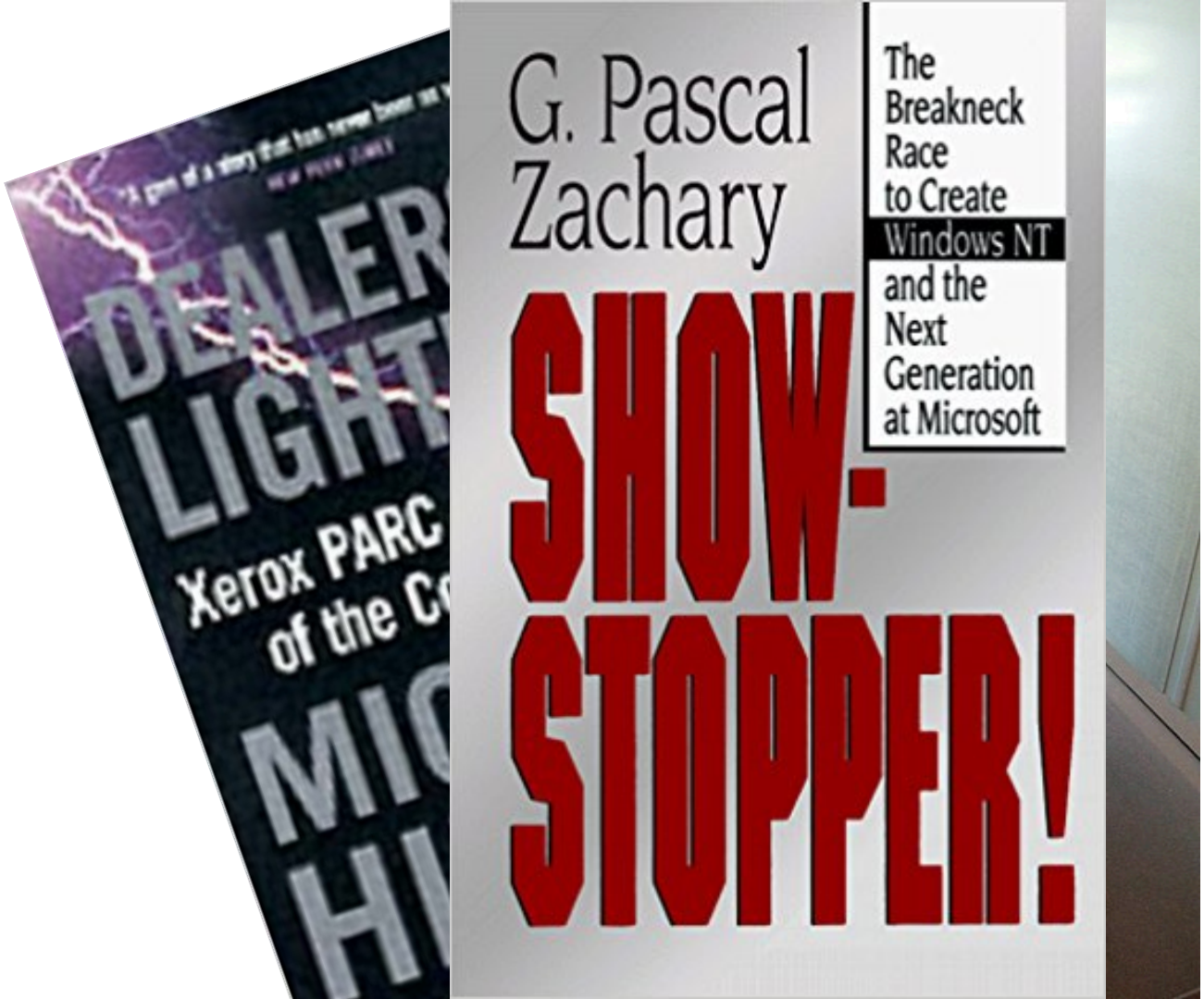
from unreliable rpc to resilience with **Linkerd**

Edward Wilde

3 April, 2018

FORM3
FINANCIAL CLOUD


FORM3
FINANCIAL CLOUD


DEALERS OF LIGHT
Xerox PARC
of the Co
MIC
HI
G. Pascal Zachary
SHOW-STOPPER!
The Breakneck Race to Create Windows NT and the Next Generation at Microsoft

A story about unreliable RPC

- System 1

- Increased reliability
- Reduced overall latency

But….

- Did not manage to reduce tail-latency
- Did not manage to protect services

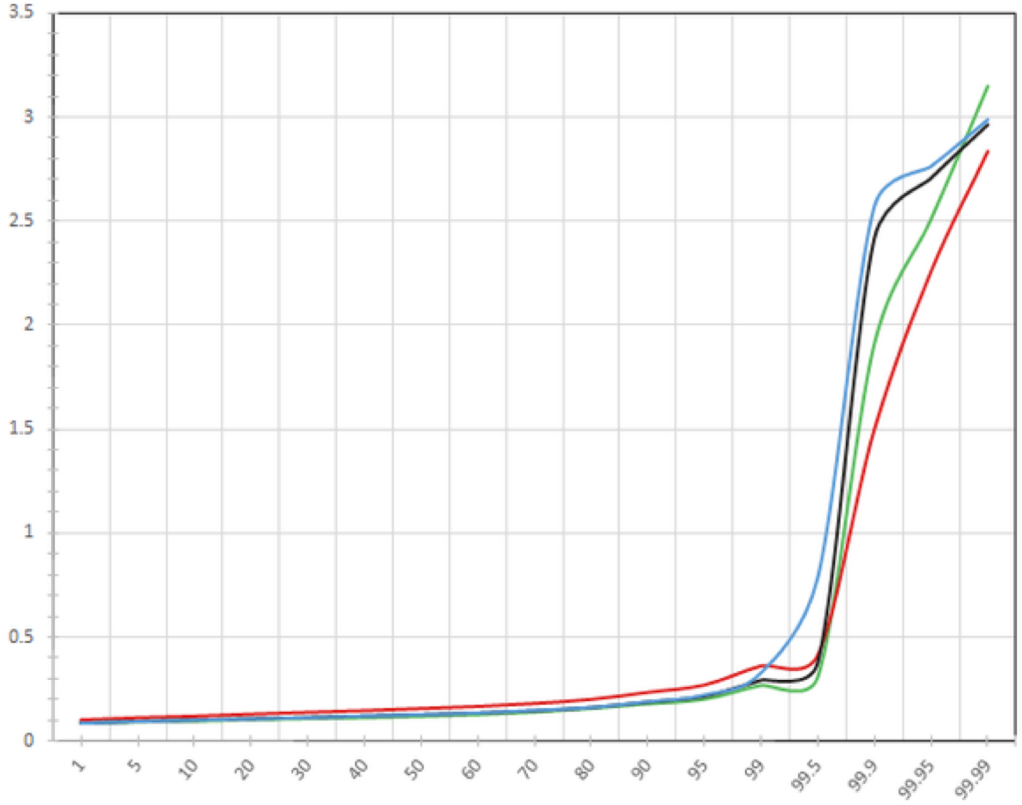
- System 2

- Increased reliability
- Reduced overall latency
- Protected services

But…

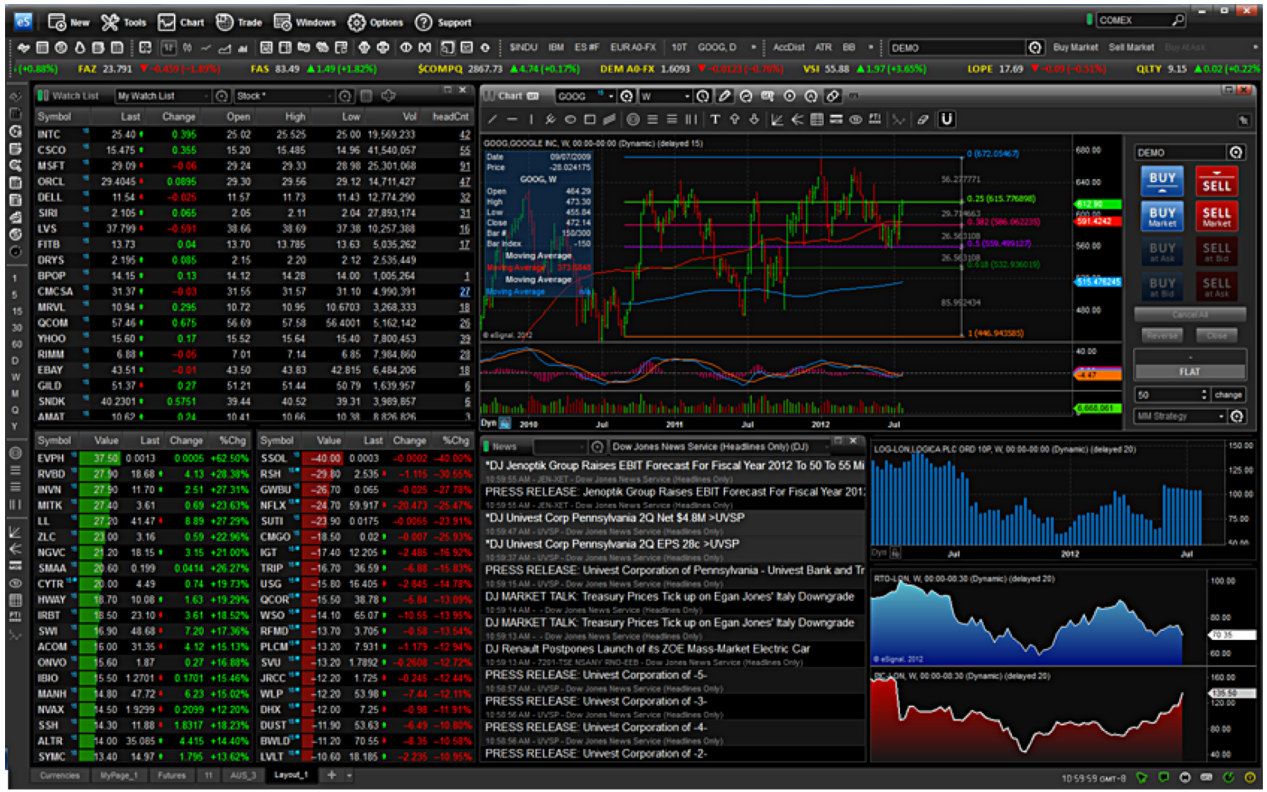
- Still need to work more on tail-latency

Tail latency

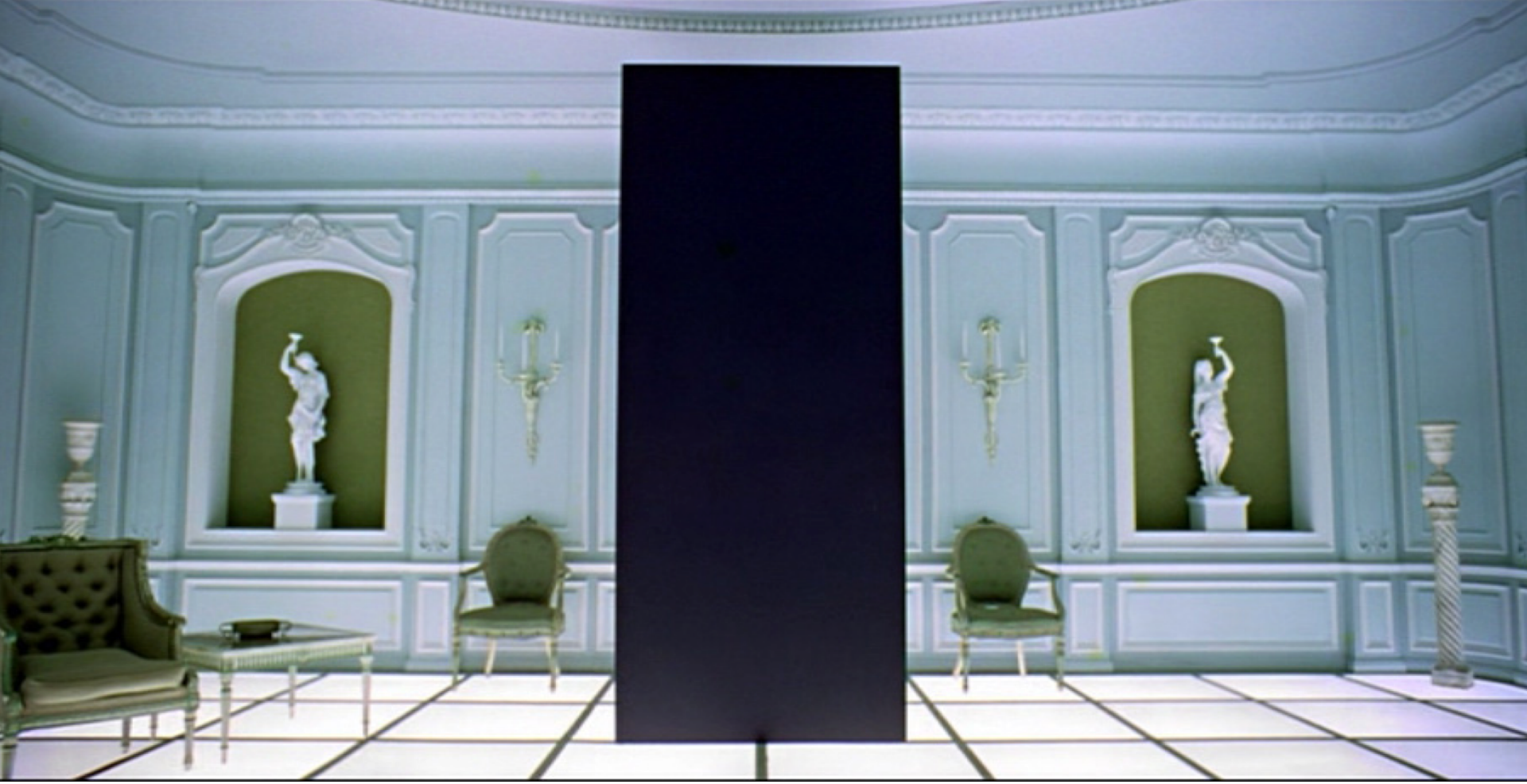


FORM3
FINANCIAL CLOUD

System 1

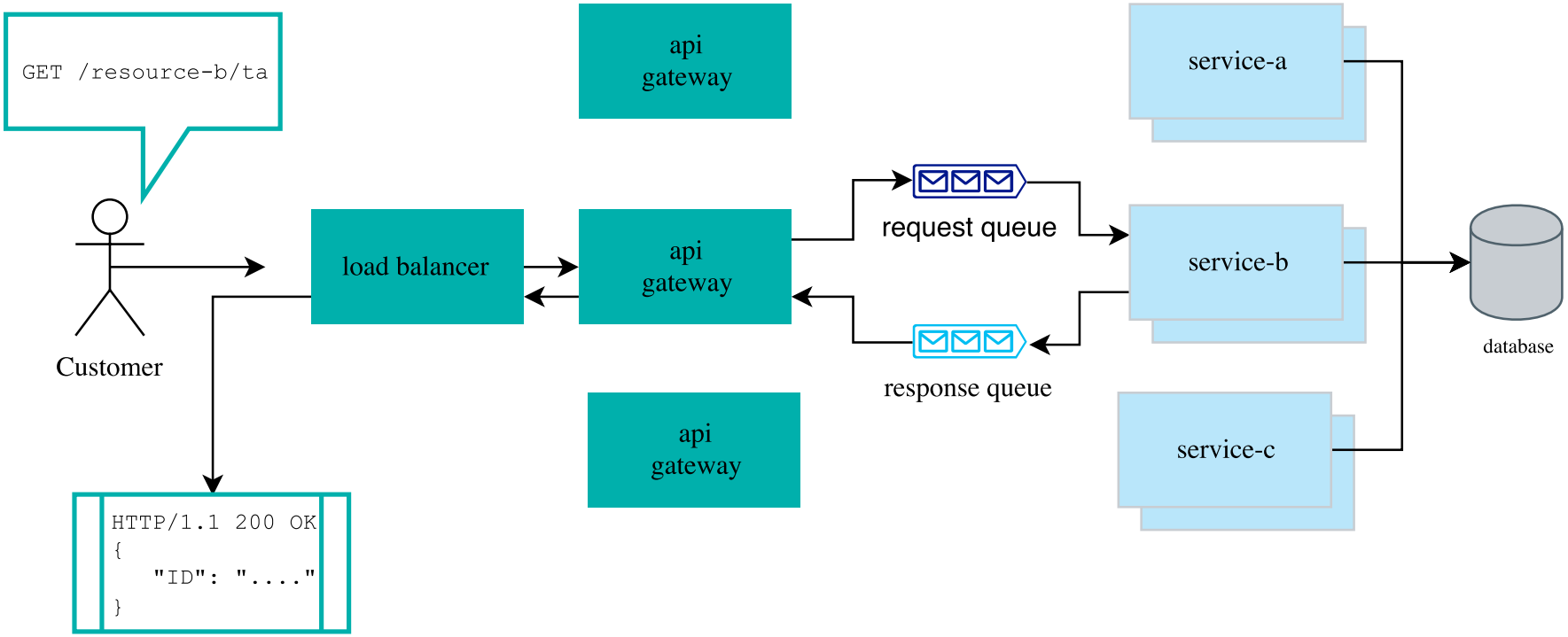


FORM3 FINANCIAL CLOUD

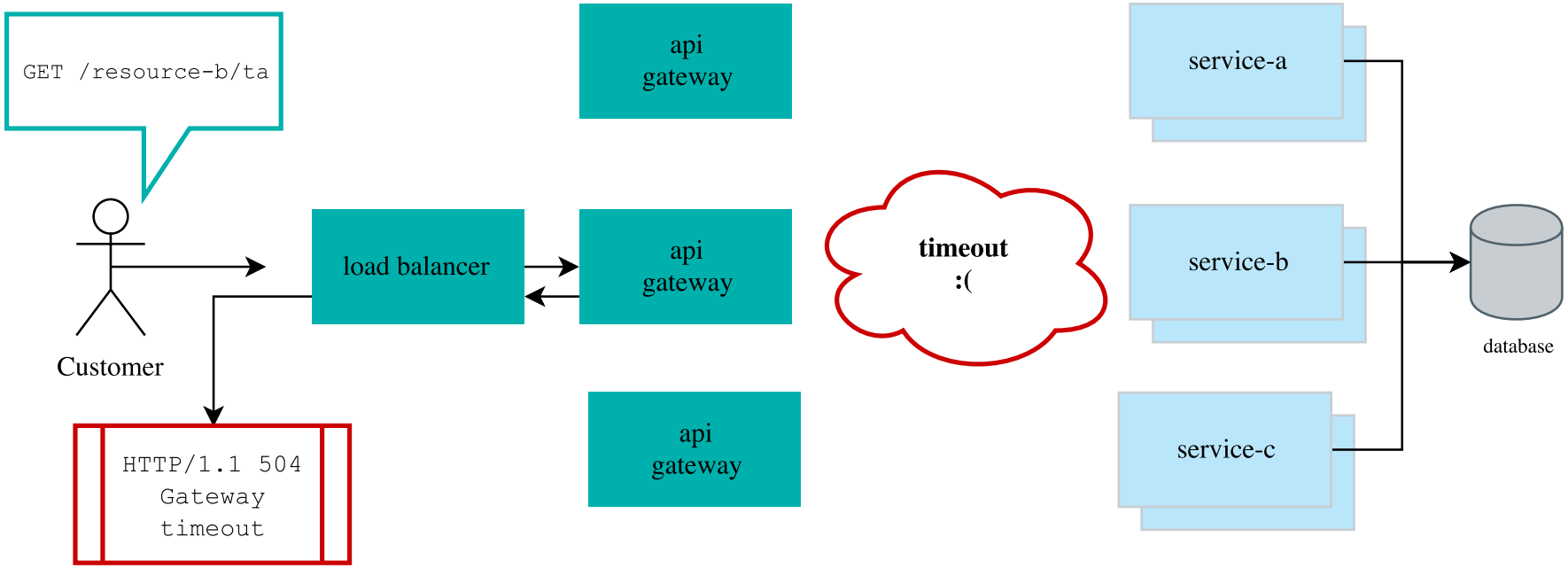


FORM3
FINANCIAL CLOUD

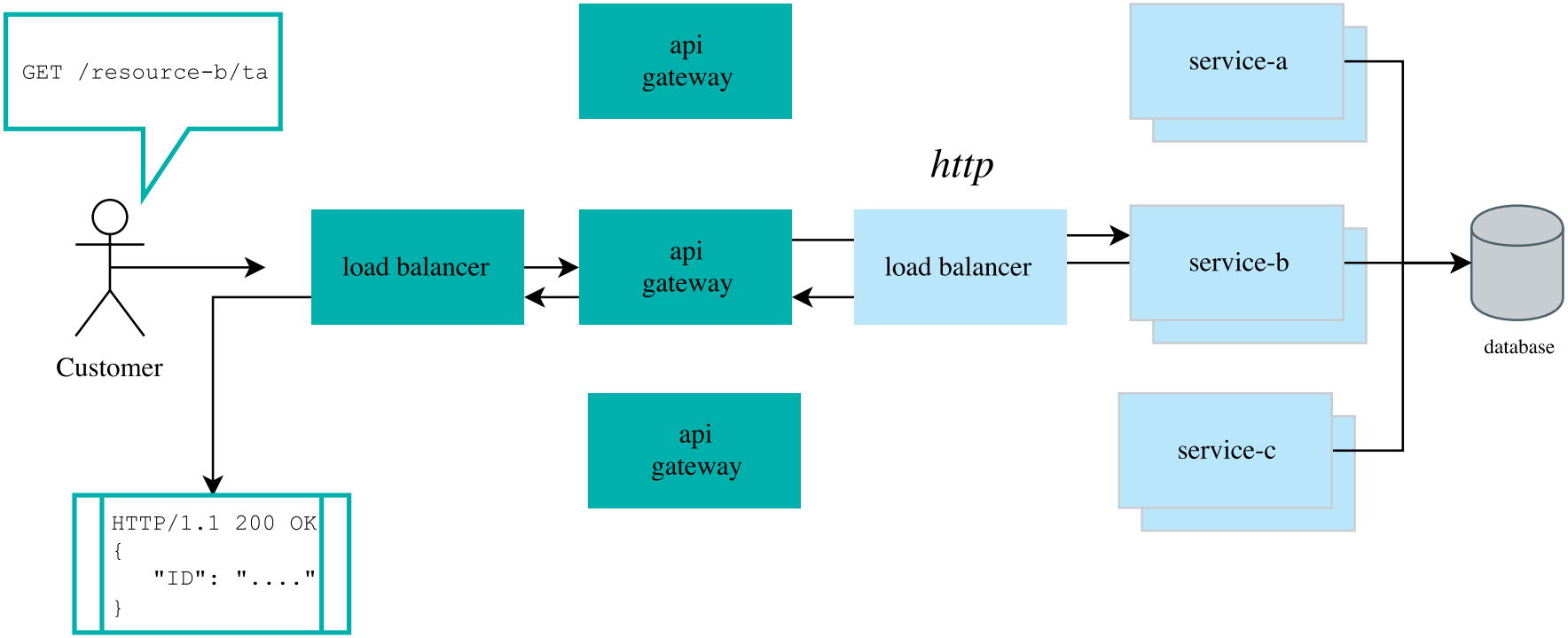
System 1



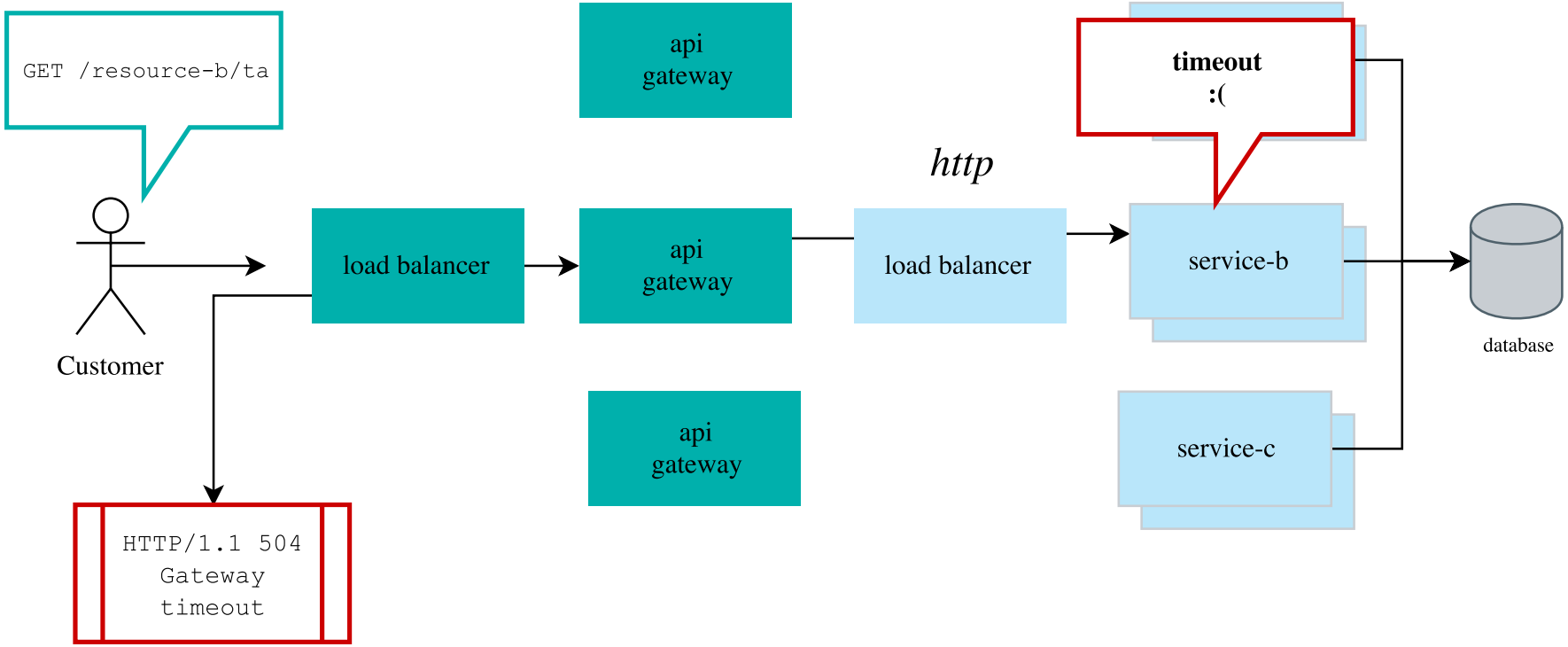
Timeout!


GET /resource-b/ta
Customer
load balancer
api gateway
api gateway
api gateway
HTTP/1.1 504 Gateway timeout
timeout :(
service-a
service-b
service-c
database

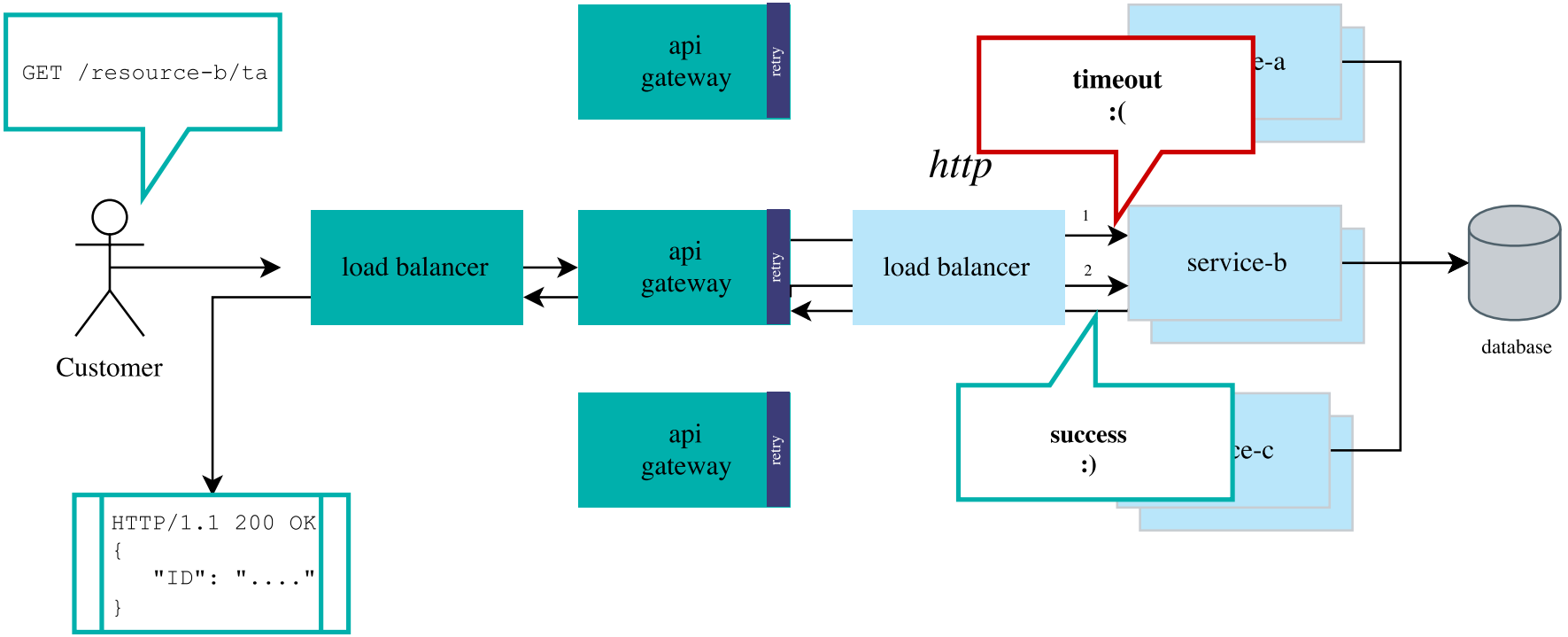
Let’s try http



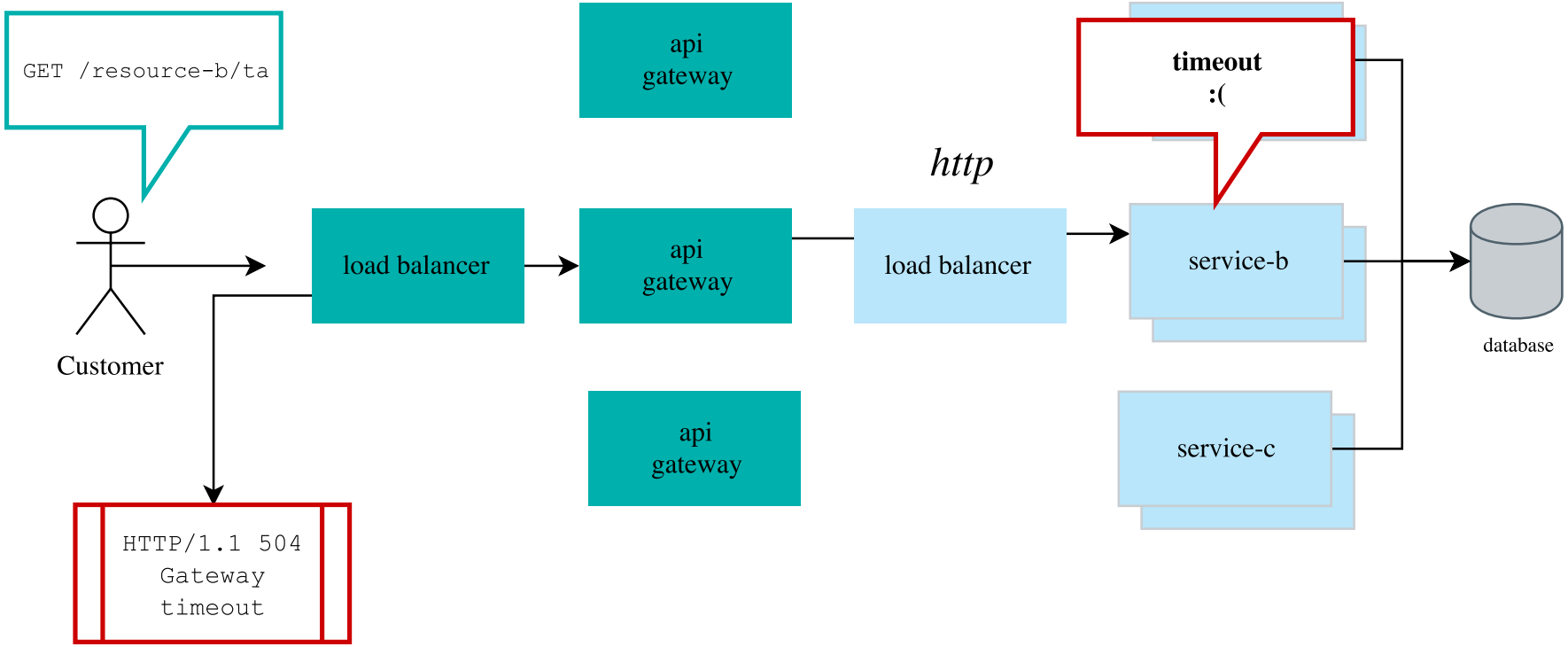
Oops still timeout



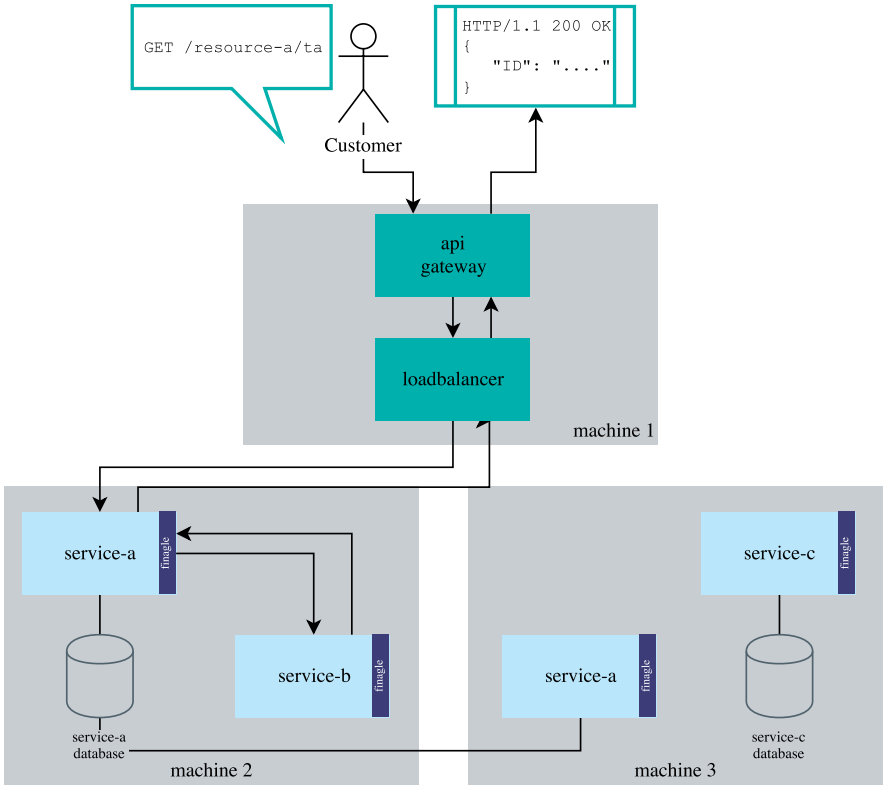
Try that again



Busy day, more timeout



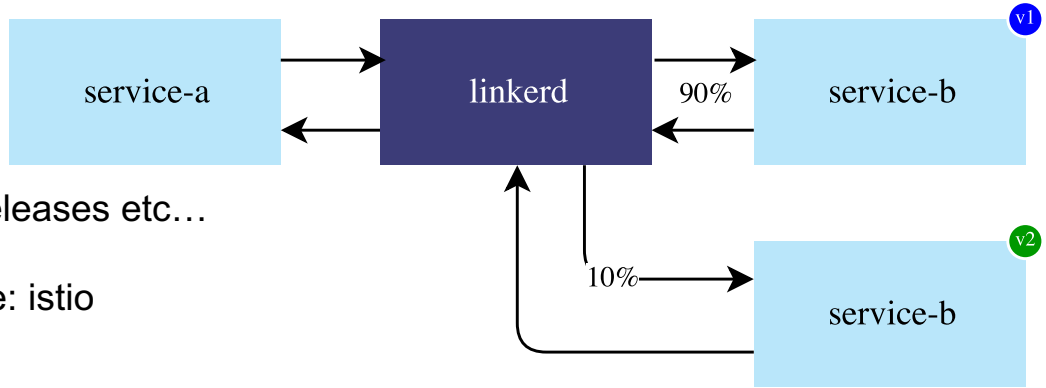
System 2: Service to service communication in a microservices architecture



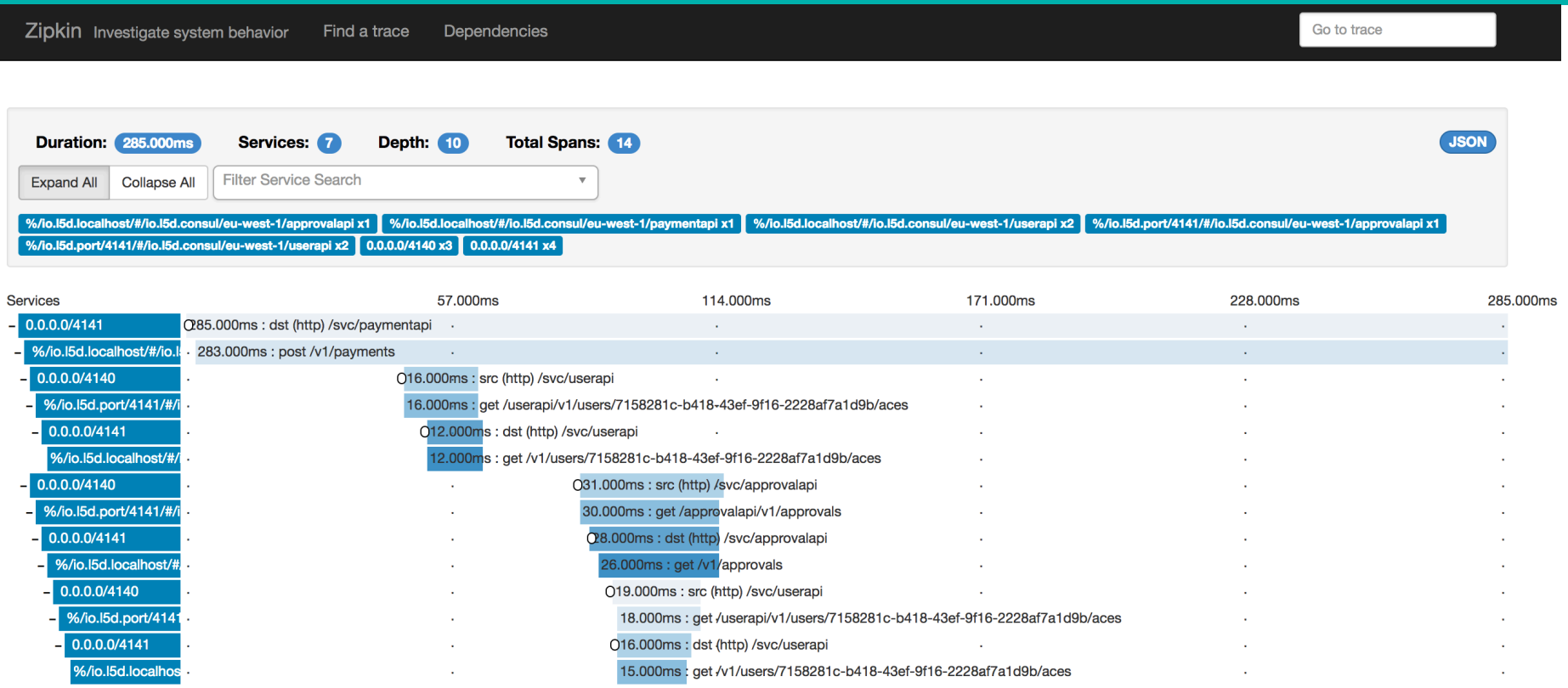
FORM3 FINANCIAL CLOUD

Service mesh

- Retry failures / timeout
- Implements circuit breaker
- Security
- Routing logic & service discovery
- Blue / green deployments: canary releases etc…
- Distributed Tracing
- Other services meshes are available: istio

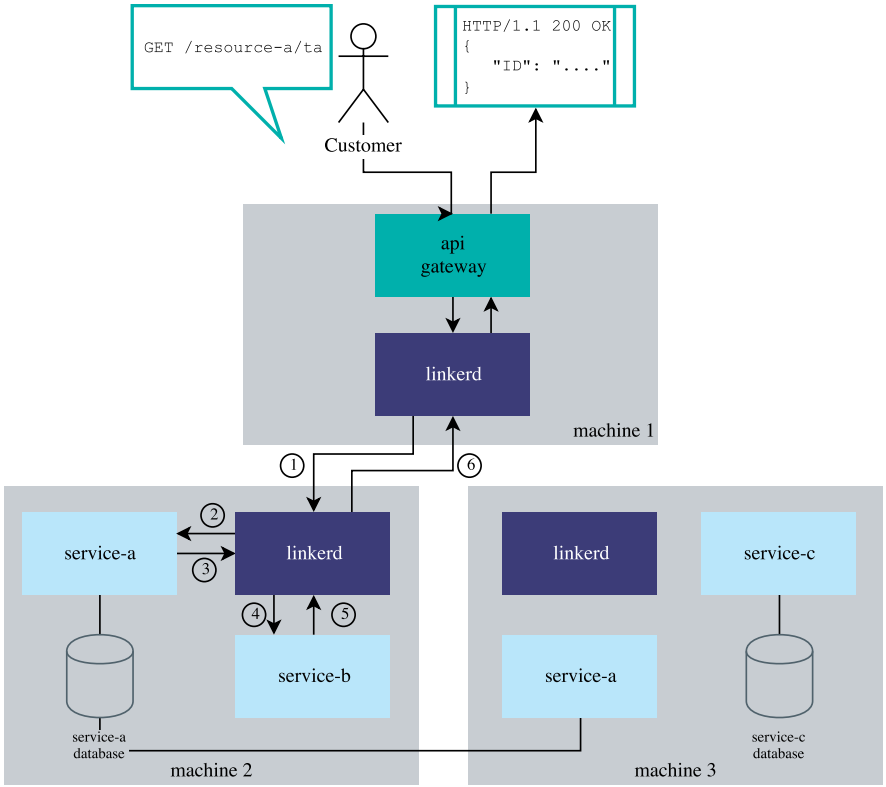


Tracing



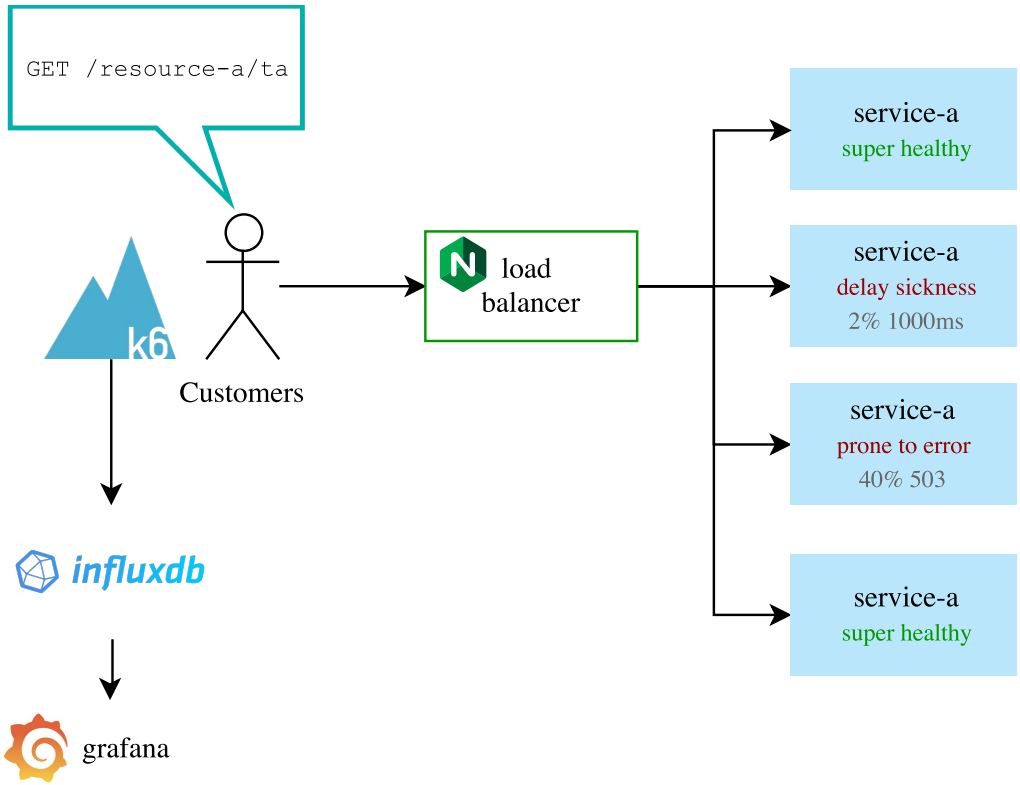
FORM3
FINANCIAL CLOUD

Service mesh in a microservices architecture



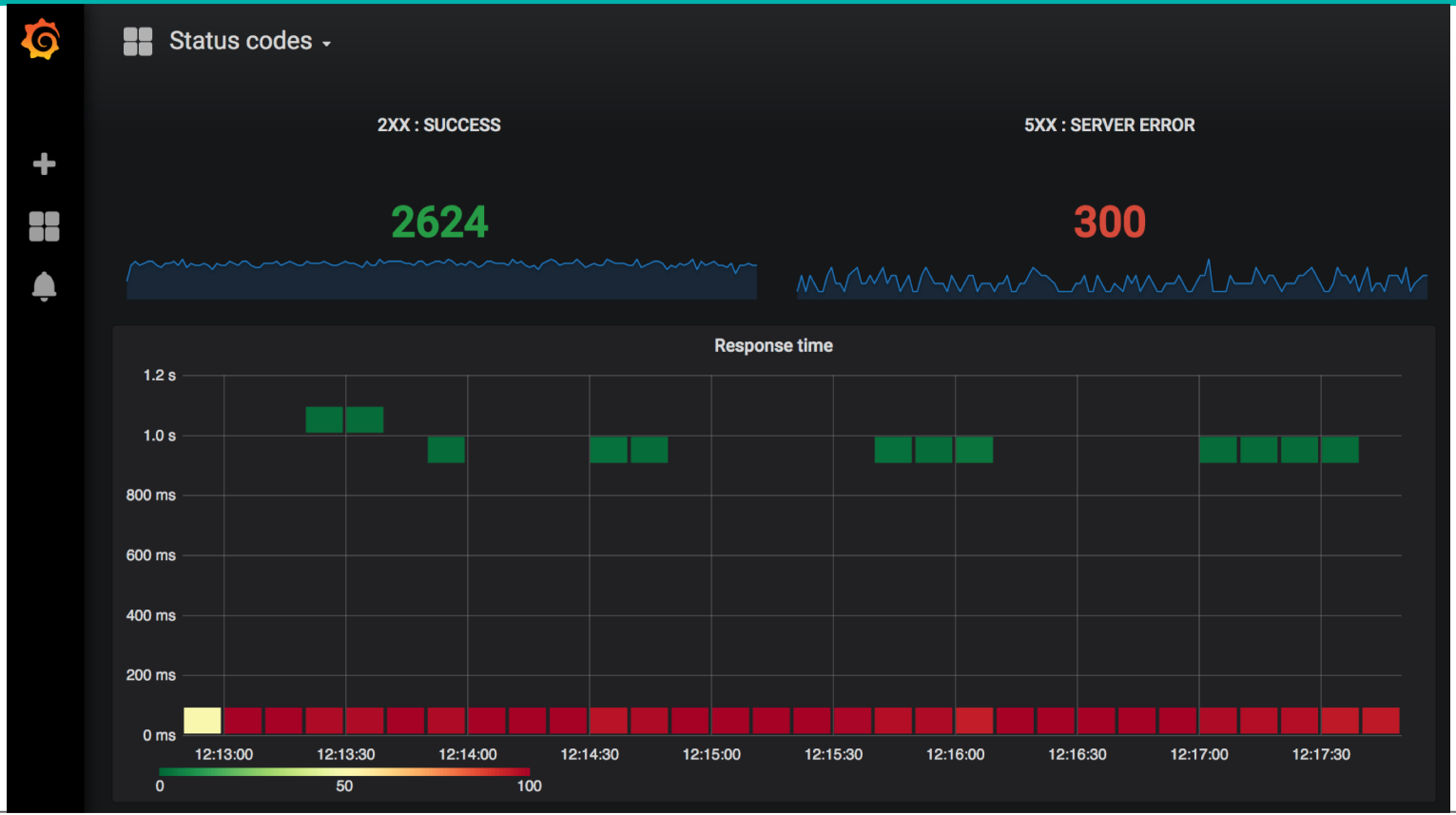
FORM3 FINANCIAL CLOUD

Demo: system-1



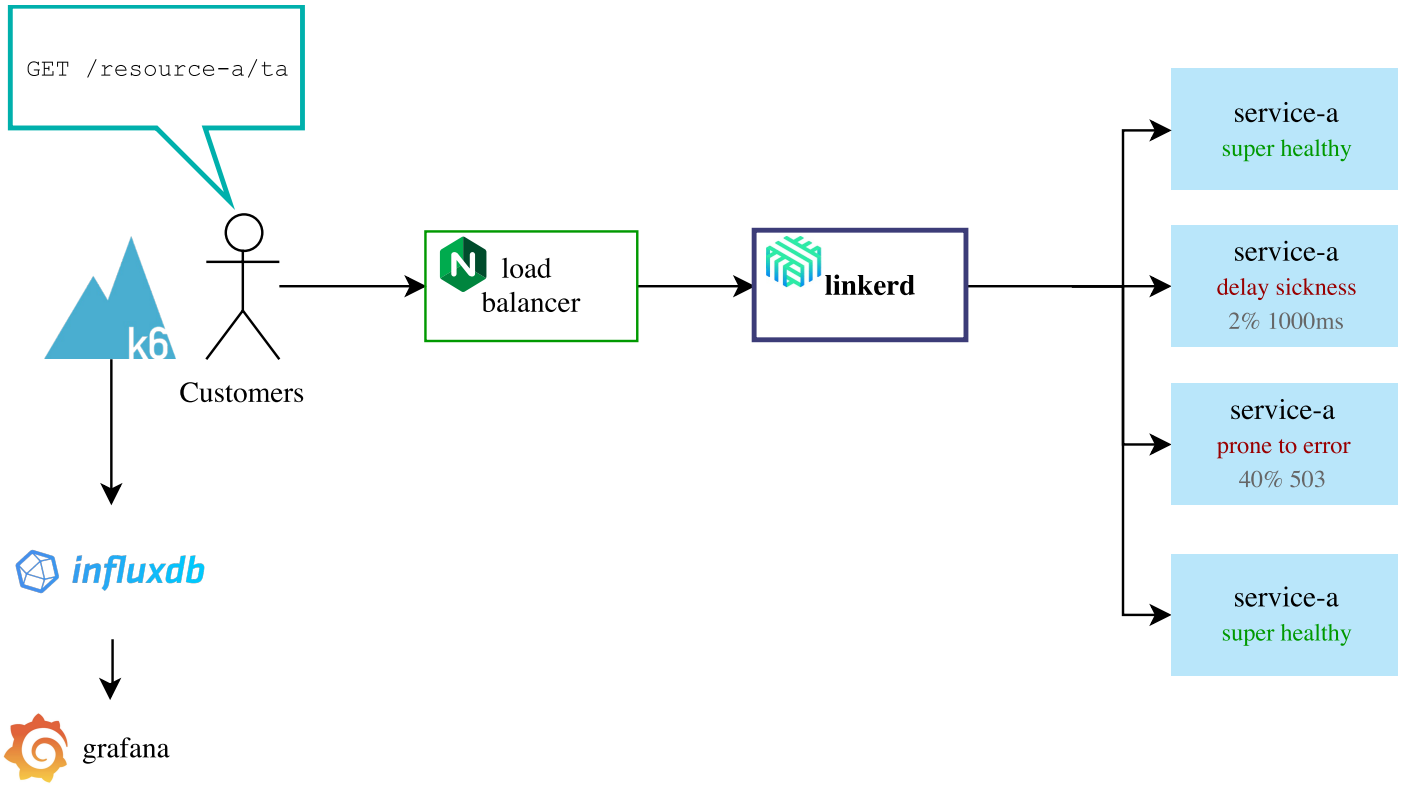
System 1: Response times

System 1: Response times



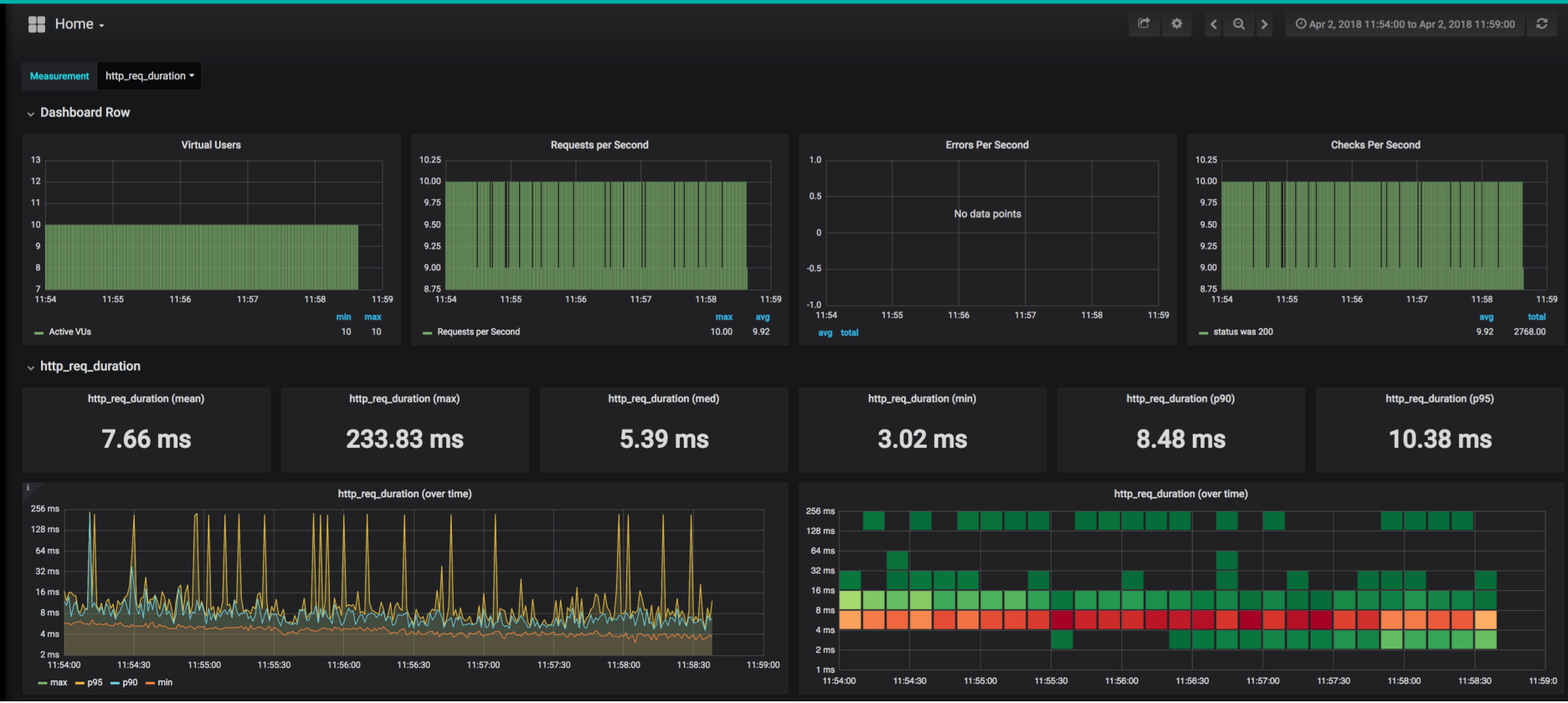
FORM3 FINANCIAL CLOUD

Demo: system-2

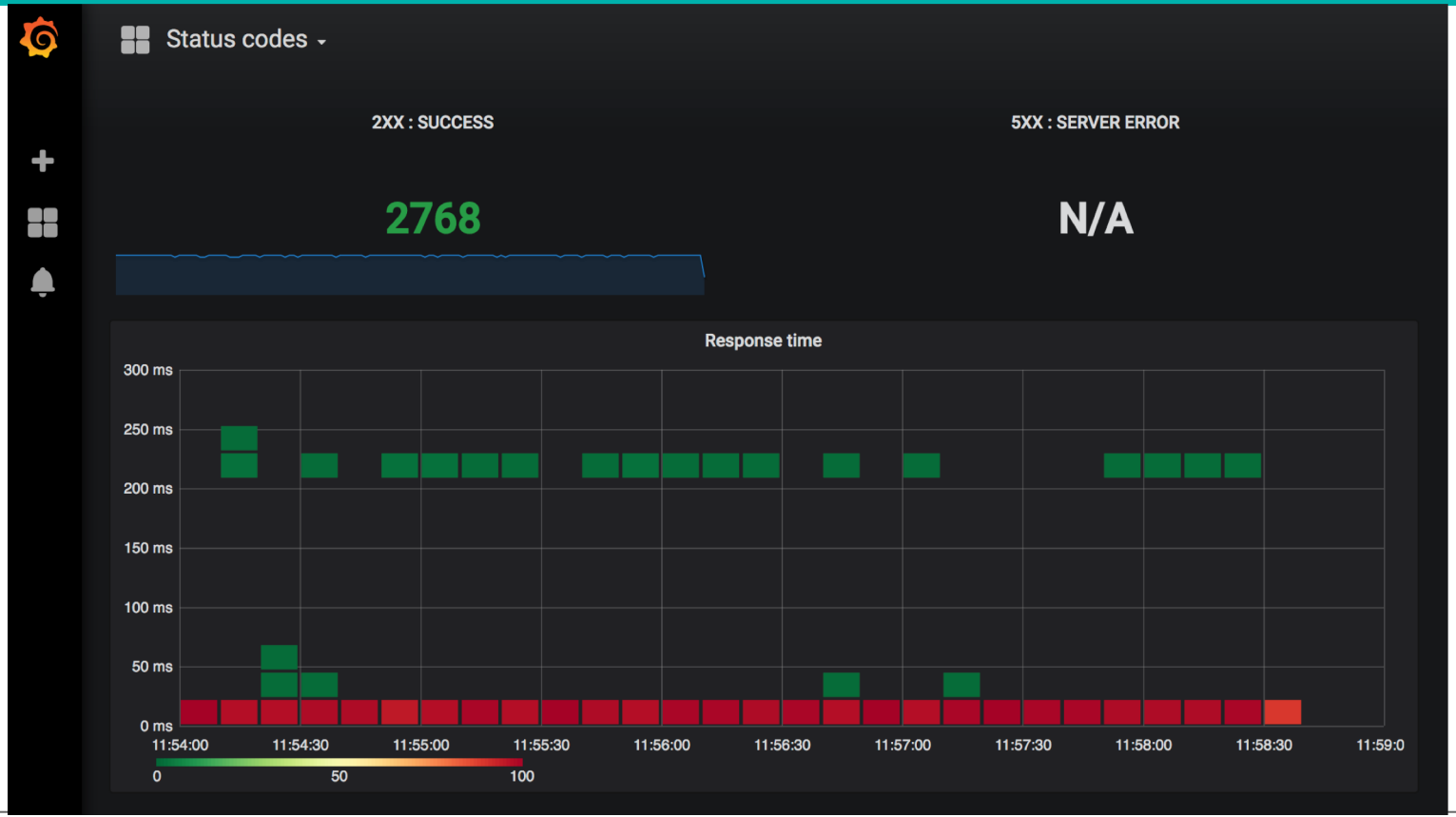


FORM3
FINANCIAL CLOUD

System 2: Response times



System 2: Response times



FORM3 FINANCIAL CLOUD

Further improving tail latency

DOI:10.1145/2408776.2408794

Software techniques that tolerate latency variability are vital to building responsive large-scale Web services.

BY JEFFREY DEAN AND LUIZ ANDRÉ BARROSO

The Tail at Scale

as overall use increases. Temporary high-latency episodes (unimportant in moderate-size systems) may come to dominate overall service performance at large scale. Just as fault-tolerant computing aims to create a reliable whole out of less-reliable parts, large online services need to create a predictably responsive whole out of less-predictable parts; we refer to such systems as "latency tail-tolerant," or simply "tail-tolerant." Here, we outline some common causes for high-latency episodes in large online services and describe techniques that reduce their severity or mitigate their effect on whole-system performance. In many cases, tail-tolerant techniques can take advantage of resources already deployed to achieve fault-tolerance, resulting in low additional overhead. We explore how these techniques allow system utilization to be driven higher without lengthening the latency tail, thus avoiding wasteful overprovisioning.

FORM3 FINANCIAL CLOUD

Wrap it up

We have used Linkerd to

- Reduce error rates using retries
- Minimize tail latencies using timeouts

- Increase security
- Help diagnose problems using tracing

Thank you

service mesh yay!

https://github.com/ewilde/kubecon ← this talk

FORM3 FINANCIAL CLOUD https://form3.tech

working and/or

FORM3 FINANCIAL CLOUD