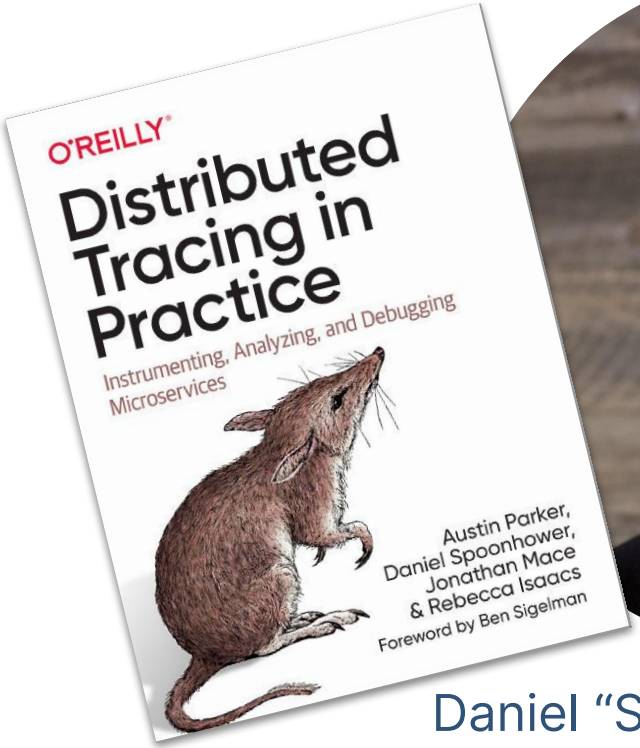
SKILUP DAYS
by: DevOps INSTITUTE

# How to Get Actual Business Value From Distributed Tracing

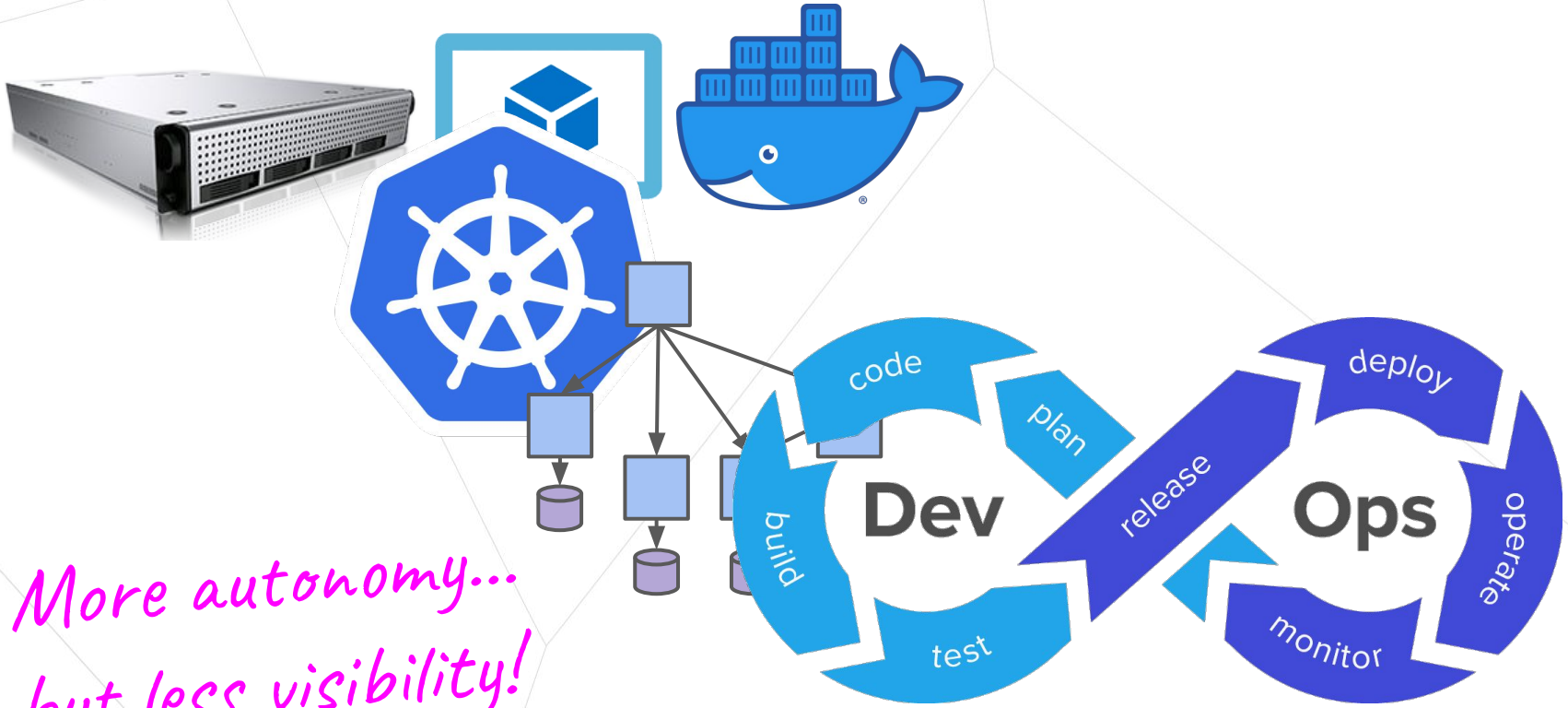
Daniel “Spoons” Spoonhower
CTO and Co-founder, Lightstep
@save_spoons
spoons@lightstep.com



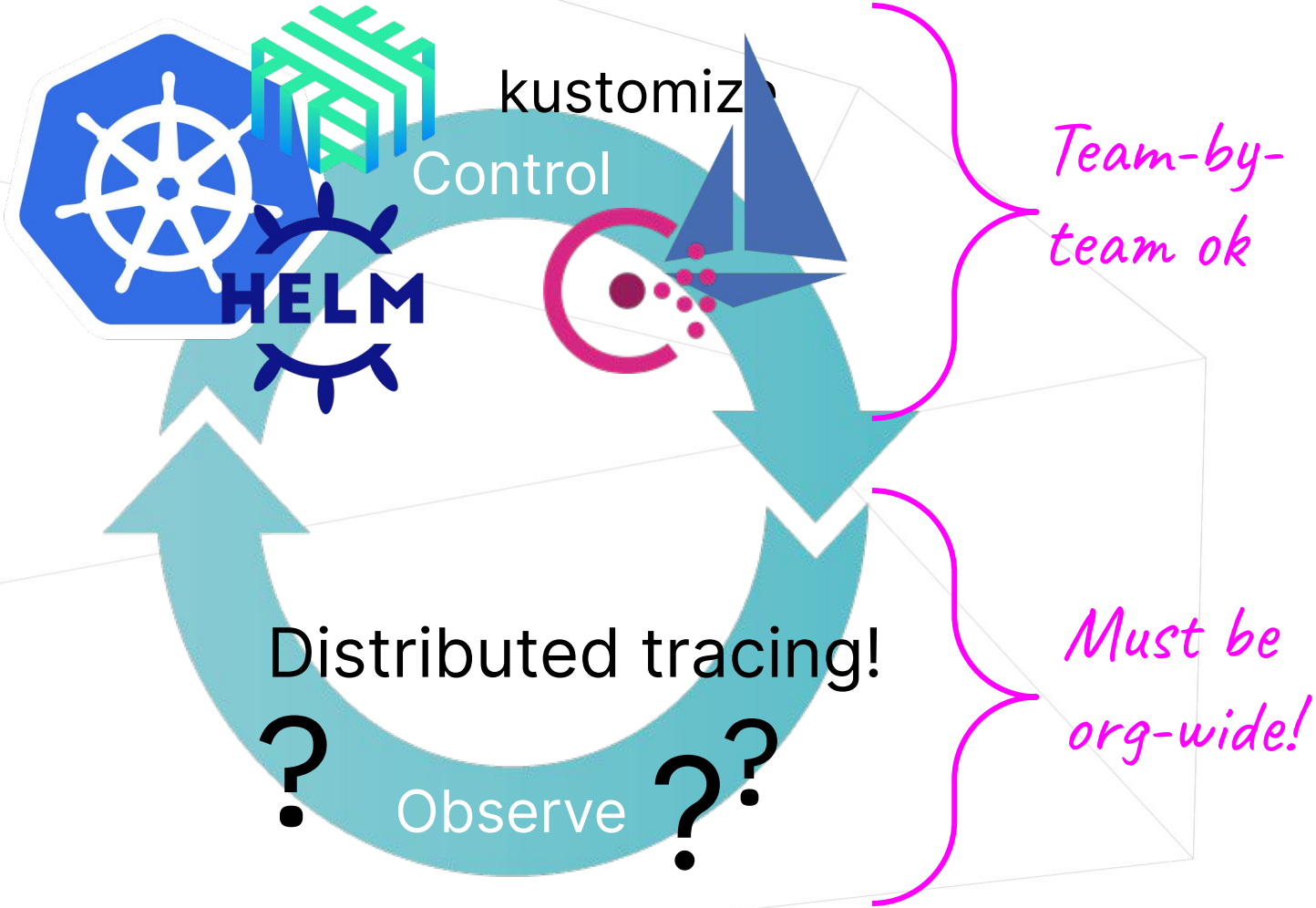

Daniel “Spoons” Spoonhower
CTO and Co-founder, Lightstep
@save_spoons

SKILUP INSTITUTE DevOps DAYS

# What Changed?




More autonomy...
but less visibility!


kustomize
Control
HELM
Distributed tracing!
?
Observe
??
Team-by-team ok
Must be org-wide!

# Observability əb-ˈzər-və-bi-lə-tē

1. *(n)* the ability to navigate **from effect to cause**
2. *(adj)* related to supporting that ability (such as a tool or process)

"used an observability tool to understand what caused the change”

For example, being able to navigate from...

Spike in errors → misconfiguration

Increased latency → new customer behavior

User complaints → upstream service deployed

# Getting Actual Business Value From Distributed tracing

Fundamentals

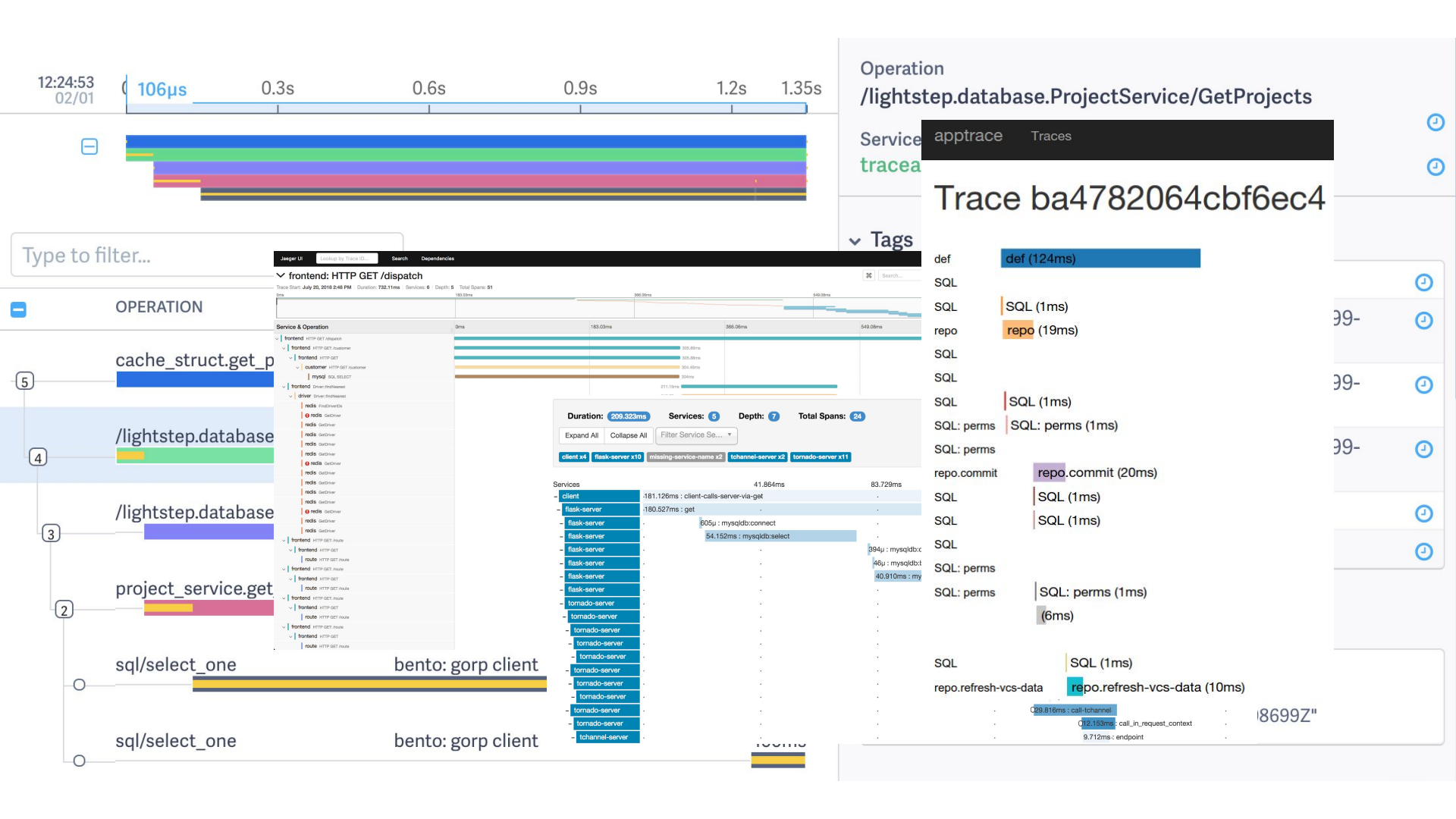
Developer velocity

Software performance

Managing costs

Deploying tracing

# Distributed Tracing


12:24:53
02/01
106µs
0.3s
0.6s
0.9s
1.2s
1.35s
Type to filter...
OPERATION
cache_struct.get_p
/lightstep.database
/lightstep.database
project_service.get
sql/select_one
bento: gorp client
sql/select_one
bento: gorp client
Operation
/lightstep.database.ProjectService/GetProjects
Service
tracea
Tags
apptrace
Traces
Trace ba4782064cbf6ec4
def
def (124ms)
SQL
SQL
SQL (1ms)
repo
repo (19ms)
SQL
SQL
SQL
SQL (1ms)
SQL: perms
SQL: perms (1ms)
SQL: perms
repo.commit
repo.commit (20ms)
SQL
SQL (1ms)
SQL
SQL (1ms)
SQL
SQL: perms
SQL: perms
SQL: perms (1ms)
(6ms)
SQL
SQL (1ms)
repo.refresh-vcs-data
repo.refresh-vcs-data (10ms)
frontend: HTTP GET /dispatch
Duration: 209.323ms
Services: 5
Depth: 7
Total Spans: 24
Expand All
Collapse All
client x4
flask-server x10
missing-service-name x2
tchannel-server x2
tornado-server x11
client
flask-server
tornado-server
tchannel-server
181.126ms : client-calls-server-via-get
180.527ms : get
605µ : mysqldb:connect
54.152ms : mysqldb:select
394µ : mysqldb:c
46µ : mysqldb:t
40.910ms : my
29.816ms : call-tchannel
12.153ms : call_in_request_context
9.712ms : endpoint
8699Z"

# Distributed tracing, defined

Traces are a form of telemetry based on *spans* with structure

- Span = timed event describing work done by a single service

Tracing is a diagnostic tool that reveals...

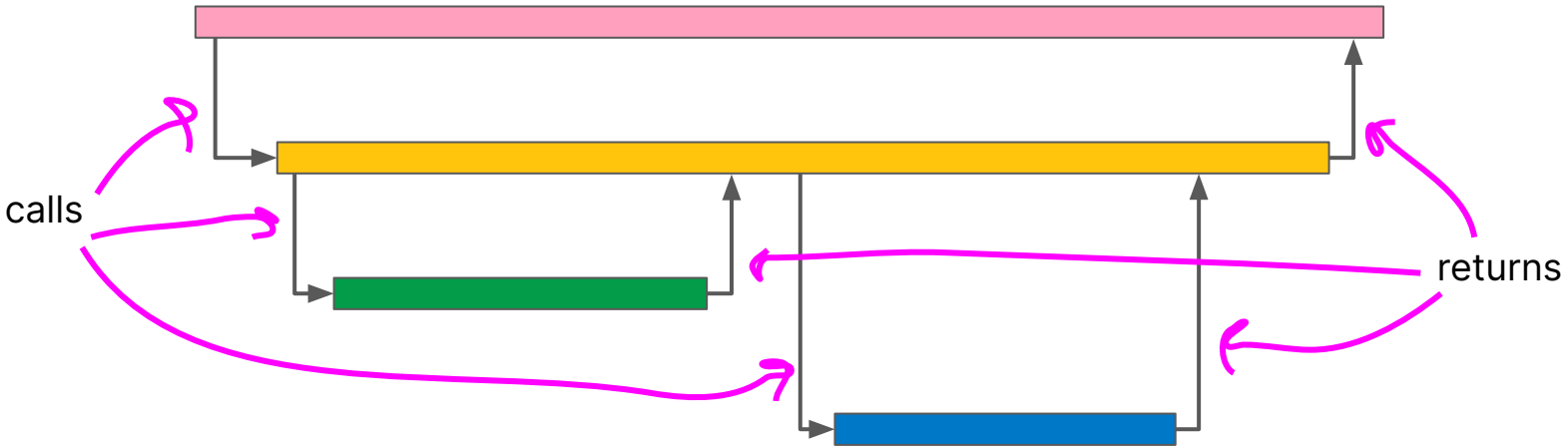
... how a set of services coordinate to handle individual user **requests**

... from mobile or browser to backends to databases (**end-to-end**)

... including metadata like **events** (logs) and **annotations** (tags)

Provides a **request-centric** view of application performance

# Relationships matter

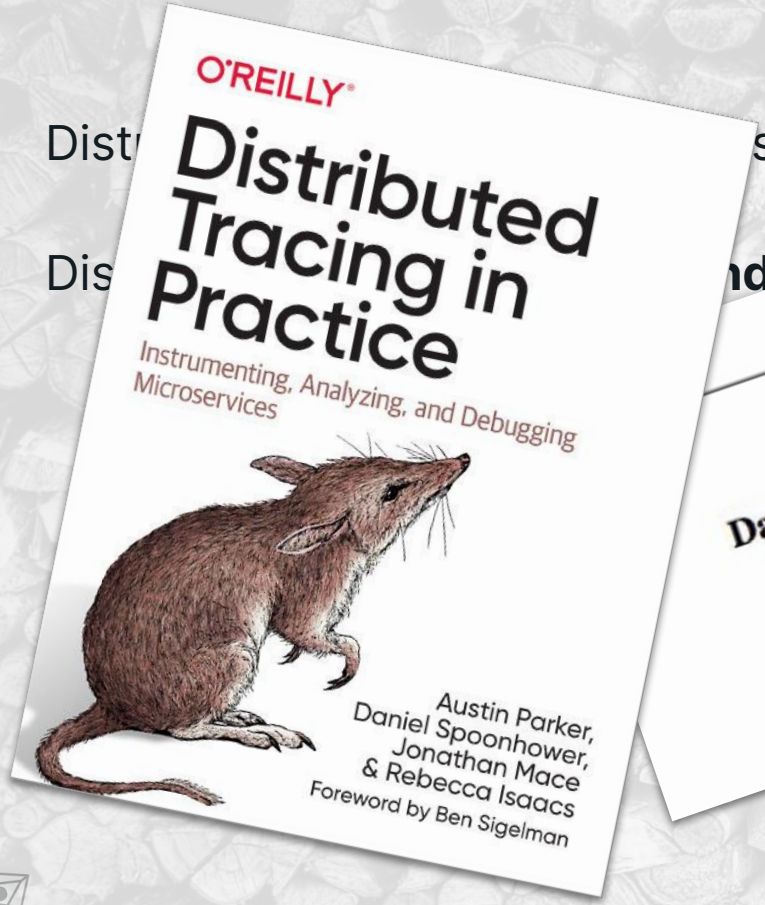


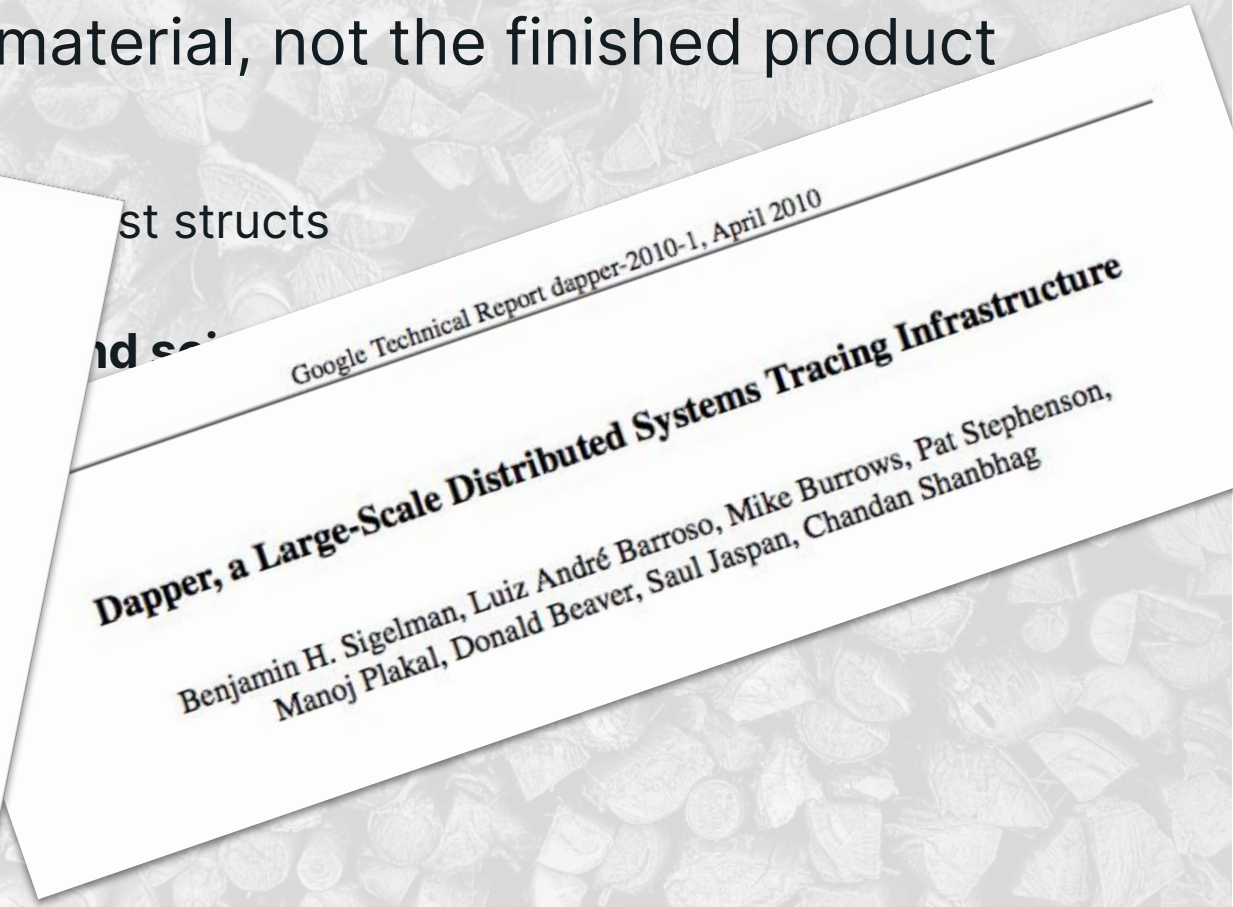

Traces encode causal relationships between callers and callees

# Traces are the raw material, not the finished product

Distr st structs

Dis nd sa



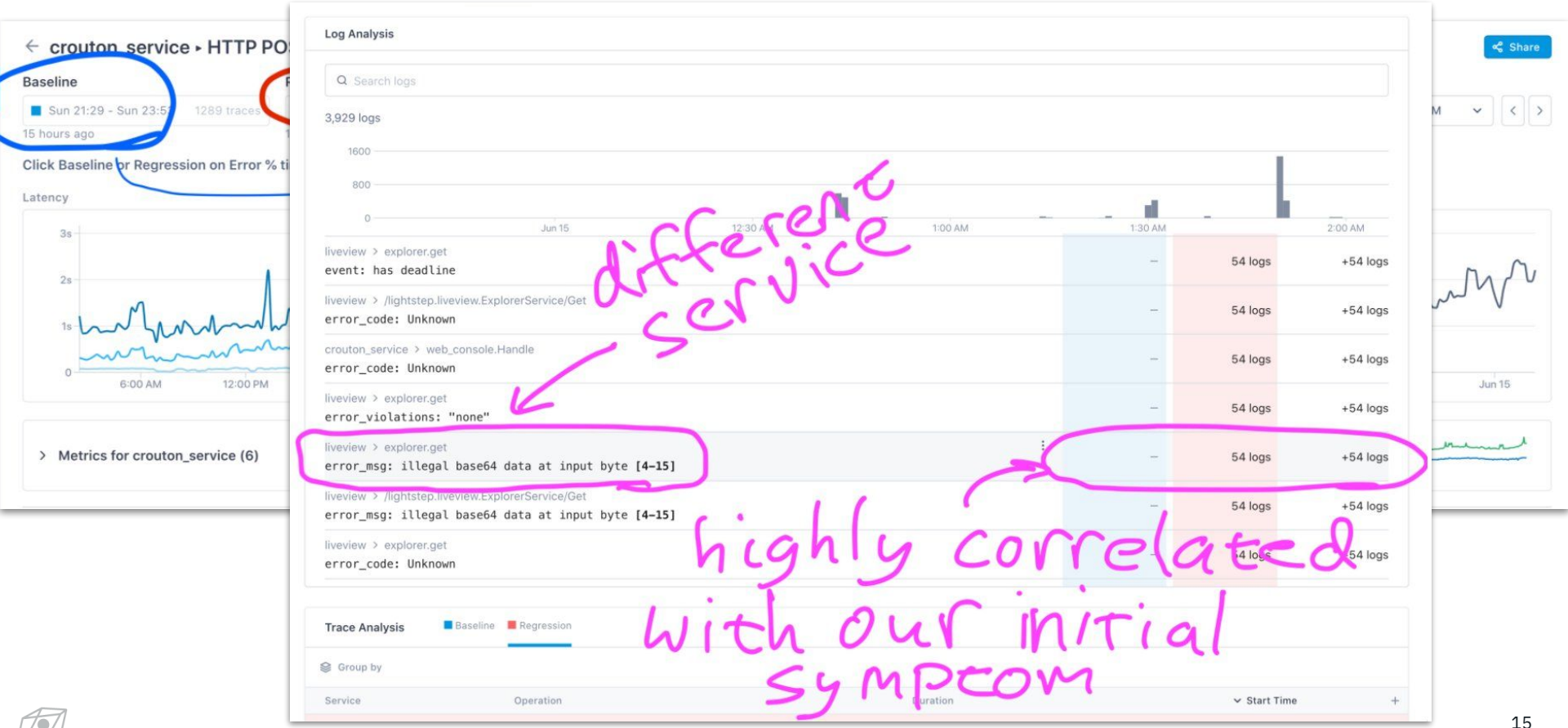



# Developer Velocity

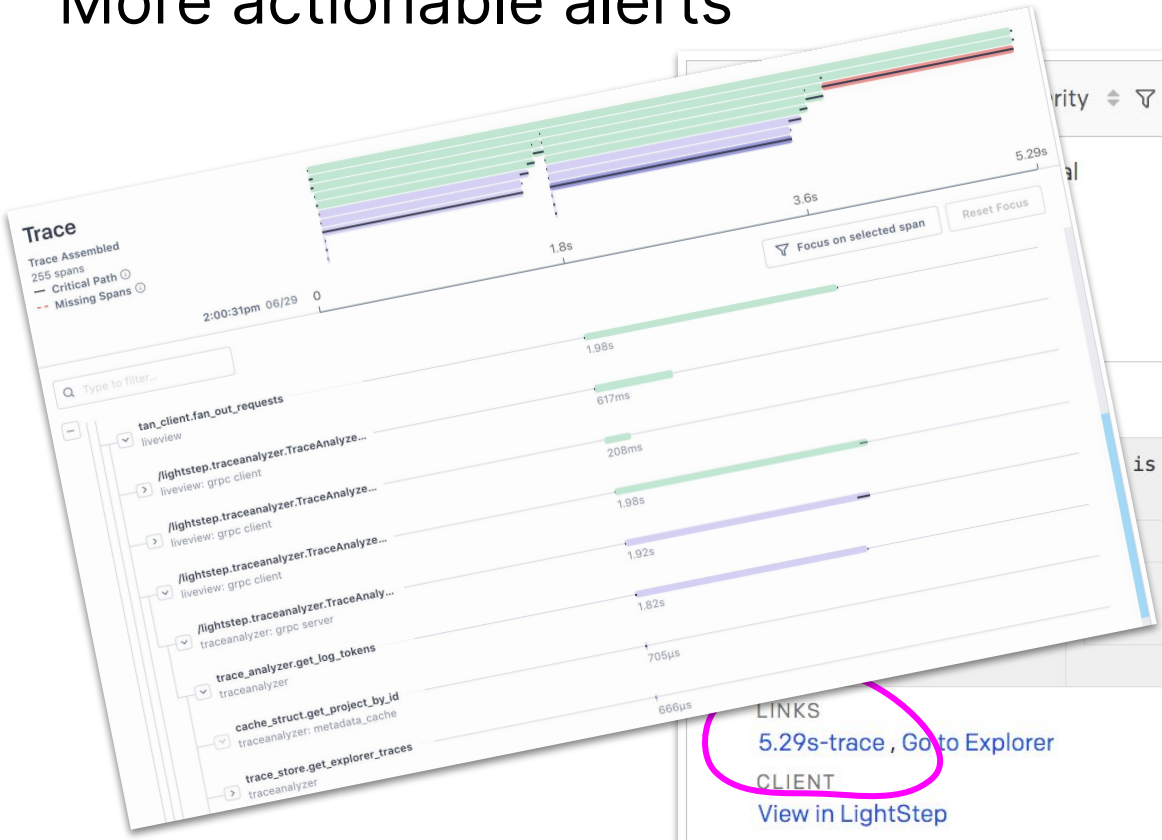
# Increasing developer velocity

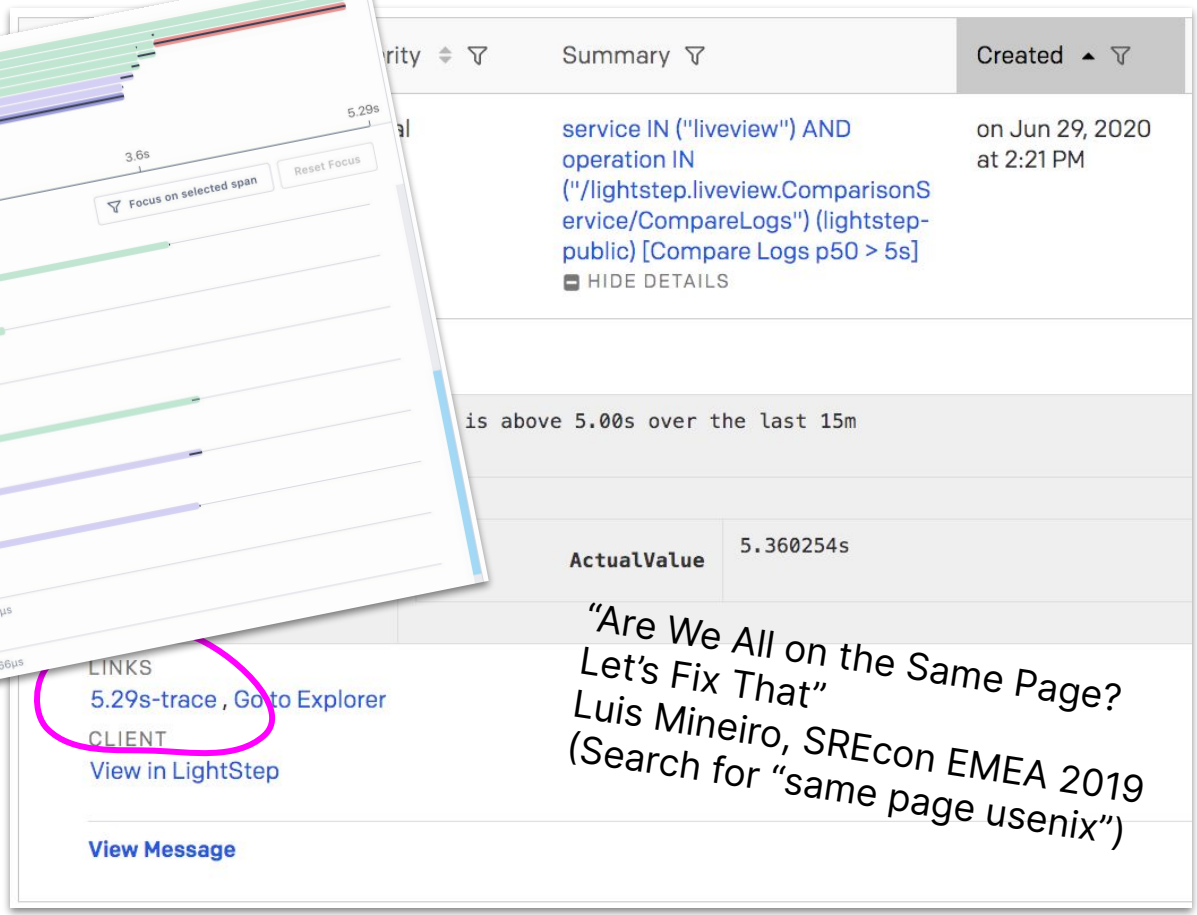
- Make (common) tasks faster
- Reduce interruptions
- Improve communication
- Prioritize high impact work

# Accelerate root cause analysis



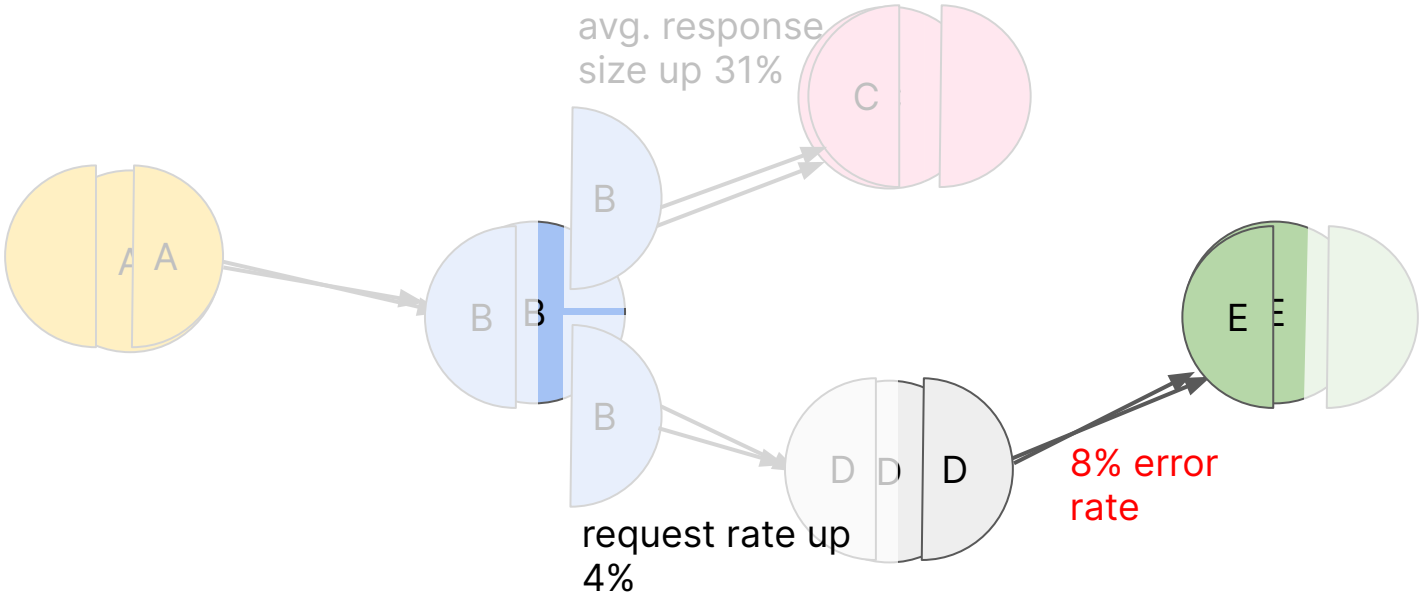
# More actionable alerts




"Are We All on the Same Page? Let's Fix That"
Luis Mineiro, SREcon EMEA 2019
(Search for "same page usenix")

# Understanding dependencies... without tracing



# Understanding dependencies

Without tracing...

- Each connection in isolation
  - “A talks to B”
- No way to narrow scope
- No way to meaningfully tie in other metrics

With tracing...

- End-to-end context
  - Request *graph*
- Can refine based on any property of the request
- Metrics linked to current scope

# Use traces and service dependencies

- Enhance training for new team members
- Facilitate operational review meetings
- Inform architectural design decisions
- Set SLOs for internal services

Use SLOs to...
- Measure reliability
- Set error budgets
- Hold teams accountable

# Software Performance

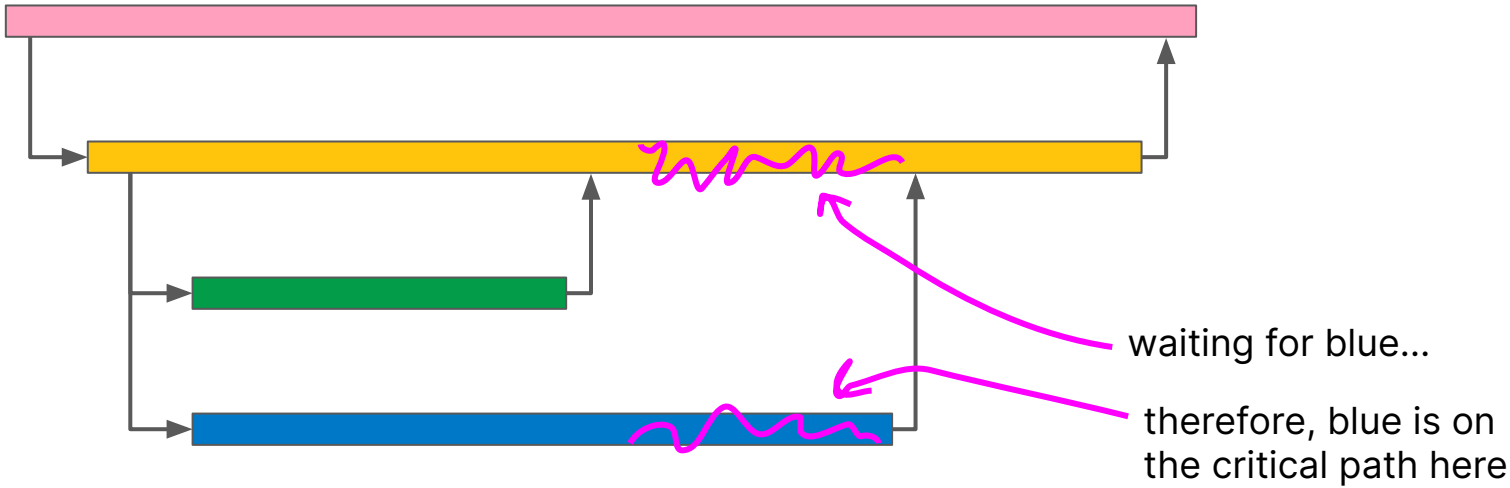
# Improving software performance

Performance means “performance as experienced by end users”

Tracing can help by...

- Better distribution of computation
- Focusing optimization where it matters

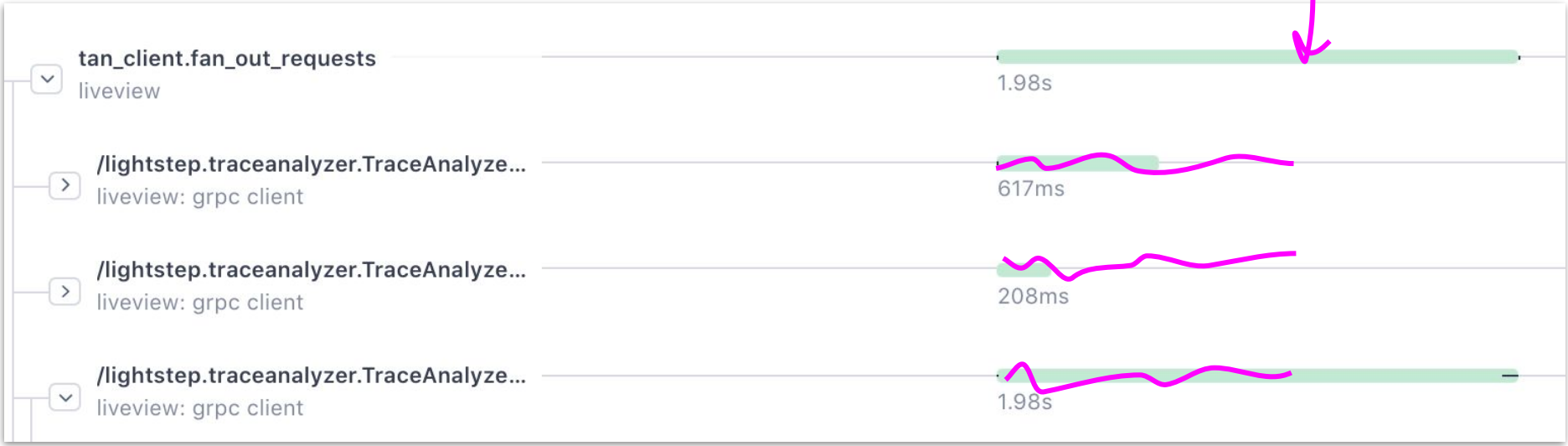
# Defining the critical path



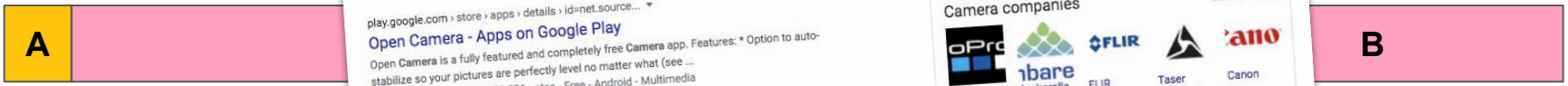

A (part of a) span is on the critical path if:

- reducing its duration speeds up overall request

# Rebalancing fan-out



# Amdahl's La




Given a choice b

1. 50% improve
2. No improvem

Obvious... once y

A

by >15%

# Managing Costs

# Types of costs

Operational costs

- Developer time (failed deployments, oncall, meeting overhead)

Revenue and reputational costs

- Missed SLOs, failed conversions, unhappy users

Infrastructure costs

- Compute, network, storage, API usage

Monitoring costs

# Calculating logging costs

Initial Factors

- Aggregating and indexing logs per service:
  - Storage
  - Compute
  - Network
- Peak instance count
- Retention period
- Services involved in a request

Initial Values

Assuming 50GB of log data a day, 14 day retention, high availability (no cold storage)

1 Primary (L Compute Optimized) @ $89

2 Data (XL Memory Optimized) @ $426

3 SSDs (General Purpose) @ $201

# $716

*Cloud spend @ 50GB/logs (monthly)*

# ~$3,386

*Total after setup, maintenance (monthly)*

# Reducing logging spend with tracing

Annotate spans with logs! It’s as easy as:

```
span.addEvent(“illegal base64 data at input byte 7”)
```

Leverage traces to determine which logs to store


Logging data is more valuable in context!

# Deploying Tracing

# On your tracing ~~migration~~ *journey*

Tracing is not an all-or-nothing endeavour

- How to deliver incremental value for the org
- How to use that value to inform next steps of the journey

Value to developers should be your (meta-)metric of success

# Step 1: Start w/ customer-critical experiences

Look at the edge and build an MVP

- As close as you can (reasonably) get to us
- Often an API gateway or proxy

Map incoming operations → dependencies

- Identify next steps
- Build a case for others to adopt tracing

# Step 2: Playbook for service owners

Establish conventions for tags, etc.

- What matters to your business?
- What would explain failures?

Instrument frameworks, libraries, shared services

- Accelerate adoption by reusing code
- Enforce conventions programmatically

# Step 3: Integrate with existing workflows

*Where* do engineers work today?

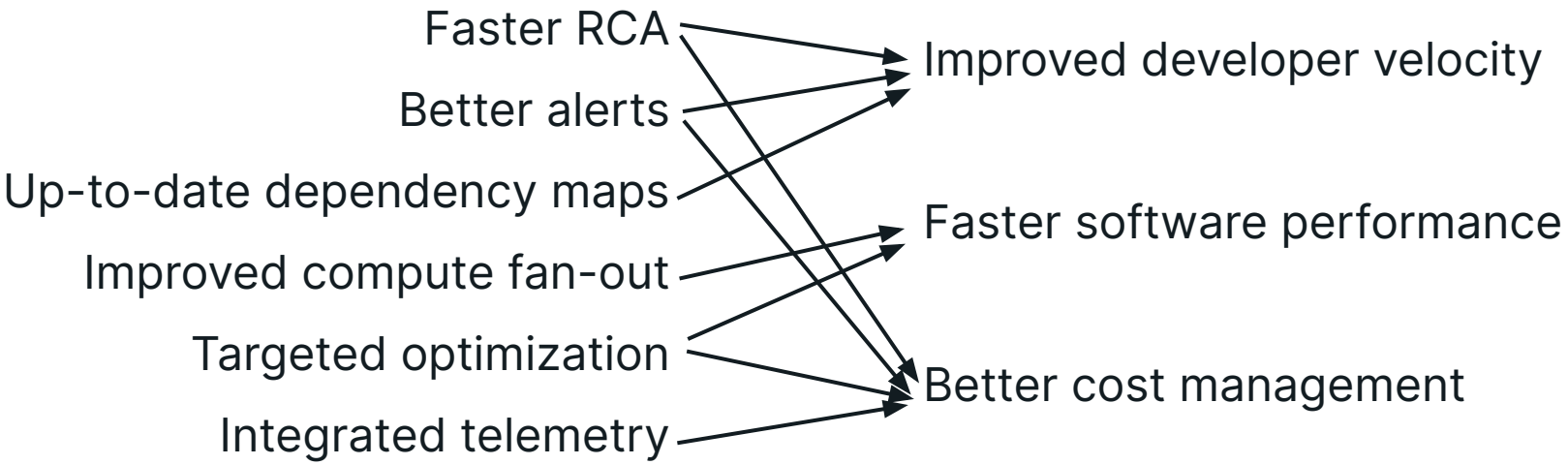
- IDEs, testing frameworks, CI/CD
- Dashboards
- Notification and alerting
- …

# Building observable services


Use open standards like OpenTelemetry for instrumenting service code.

OpenTelemetry provides a single set of APIs, SDKs, and tools for generating distributed traces and metrics from your services.

# In summary, distributed tracing provides...




*Distributed tracing puts application behavior **in context***
*to help answer the primary question of observability:*
*“What caused that change?”*

# THANK YOU!

## Meet me in the Network Chat Lounge for questions

Daniel “Spoons” Spoonhower
CTO and Co-founder, Lightstep
@save_spoons

SKILUP DAYS