RESILIENCE

REALIZED

KubeCon | CloudNativeCon

North America 2021

KubeCon | CloudNativeCon

North America 2021

RESILIENCE
REALIZED

Correlating Signals in OpenTelemetry

Jaana Dogan, Principal Engineer @ AWS
Morgan McLean, Director of PM @ Splunk

Morgan McLean
Director of PM @ Splunk

Jaana Dogan
Principal Engineer @ AWS

OpenTelemetry: Quick Intro

OpenTelemetry provides:

- A Collector (typically run as an agent / daemonset, can be run as a clustered service)
- SDKs and language auto-instrumentation agents
- A protocol
- A specification and semantic conventions for various use cases
- A community!

OpenTelemetry is opinionated: there is one set of semantic conventions for a given scenario, there is one SDK and / or language agent for a given language, etc.

OpenTelemetry: Quick Intro

OpenTelemetry captures:

- Traces (GA)
- Metrics (beta, possibly GA by the time that you’re watching this!)
- Logs (alpha, **NOT BETA**)
- Resource metadata

Background

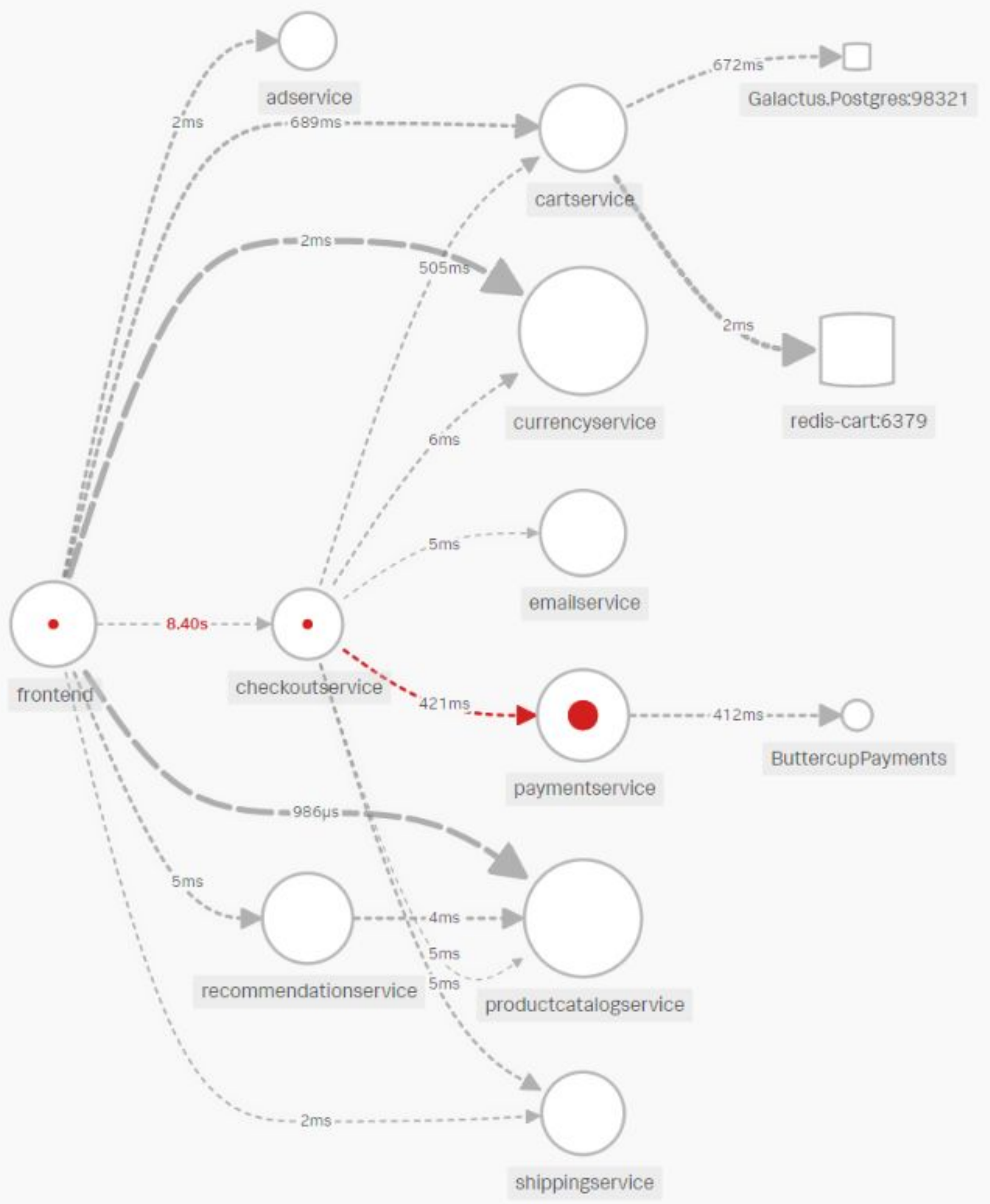
- If you don’t capture and analyze traces, metrics, logs, and other telemetry, **you are setting yourself up for failure**
- If you capture and analyze traces, metrics, logs, and other telemetry in siloes, **you are setting yourself up for failure**

WHY ARE YOU SETTING YOURSELF UP FOR FAILURE?
The point of capturing and analyzing telemetry is to give you visibility into the entire stack! Looking at pieces in isolation doesn’t achieve this.

Failures 10 levels down in a system will manifest in very strange ways at upper layers!

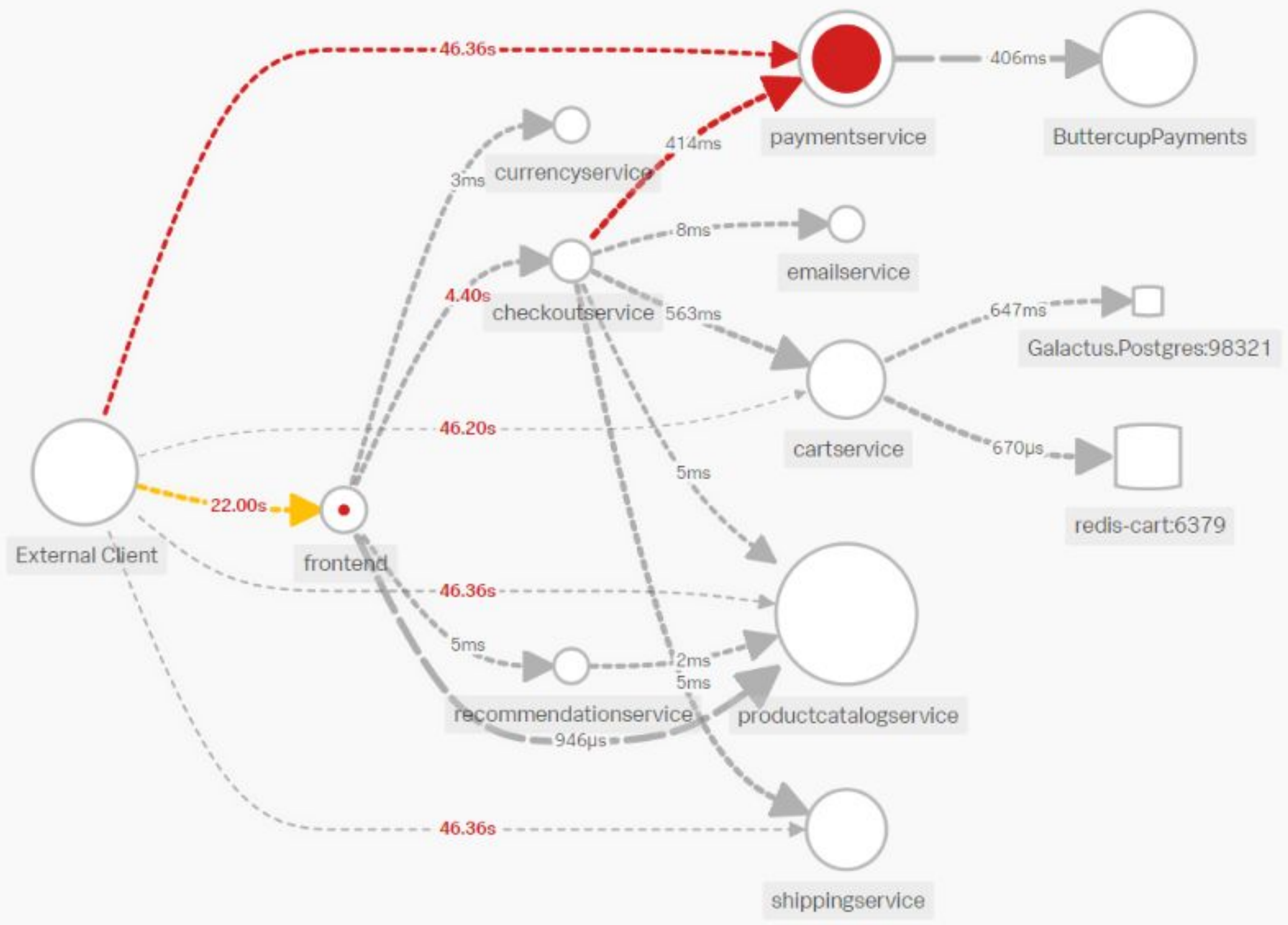
Example: Errors Causing Latency

KubeCon CloudNativeCon North America 2021

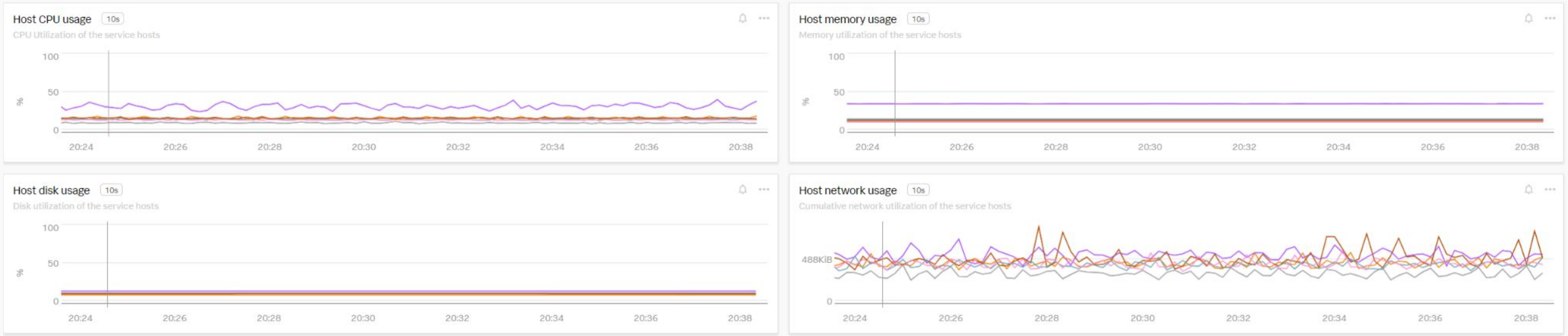


Example: Errors Causing Latency

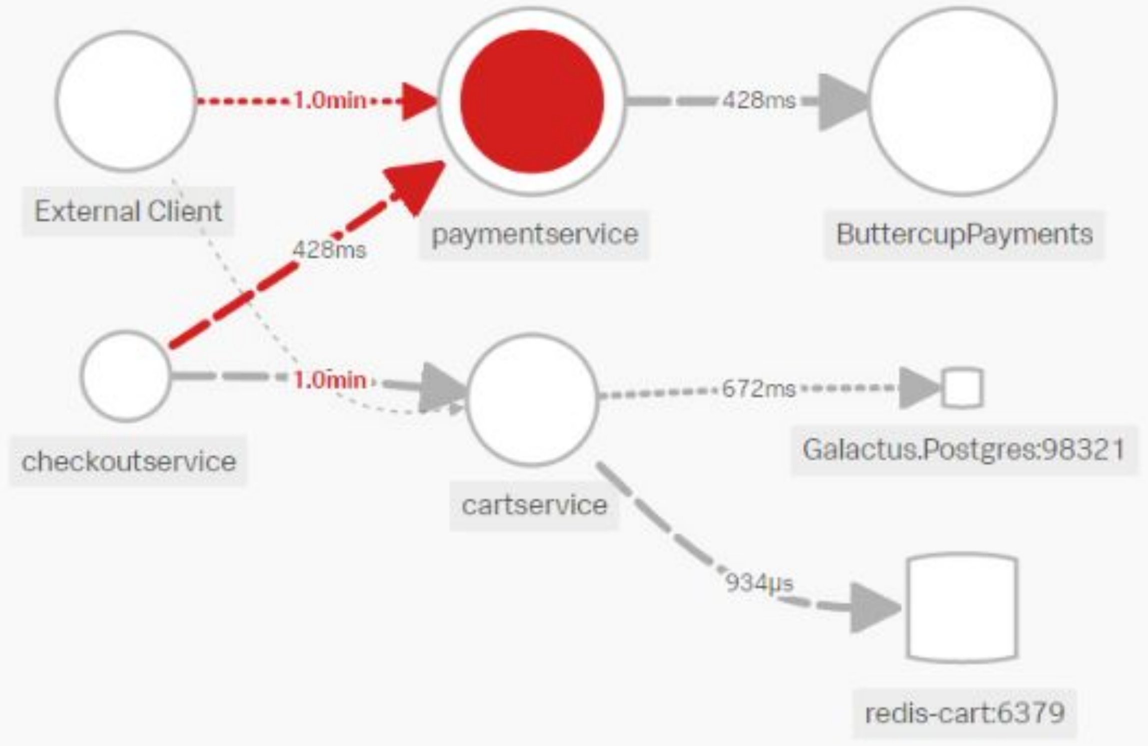
KubeCon CloudNativeCon North America 2021



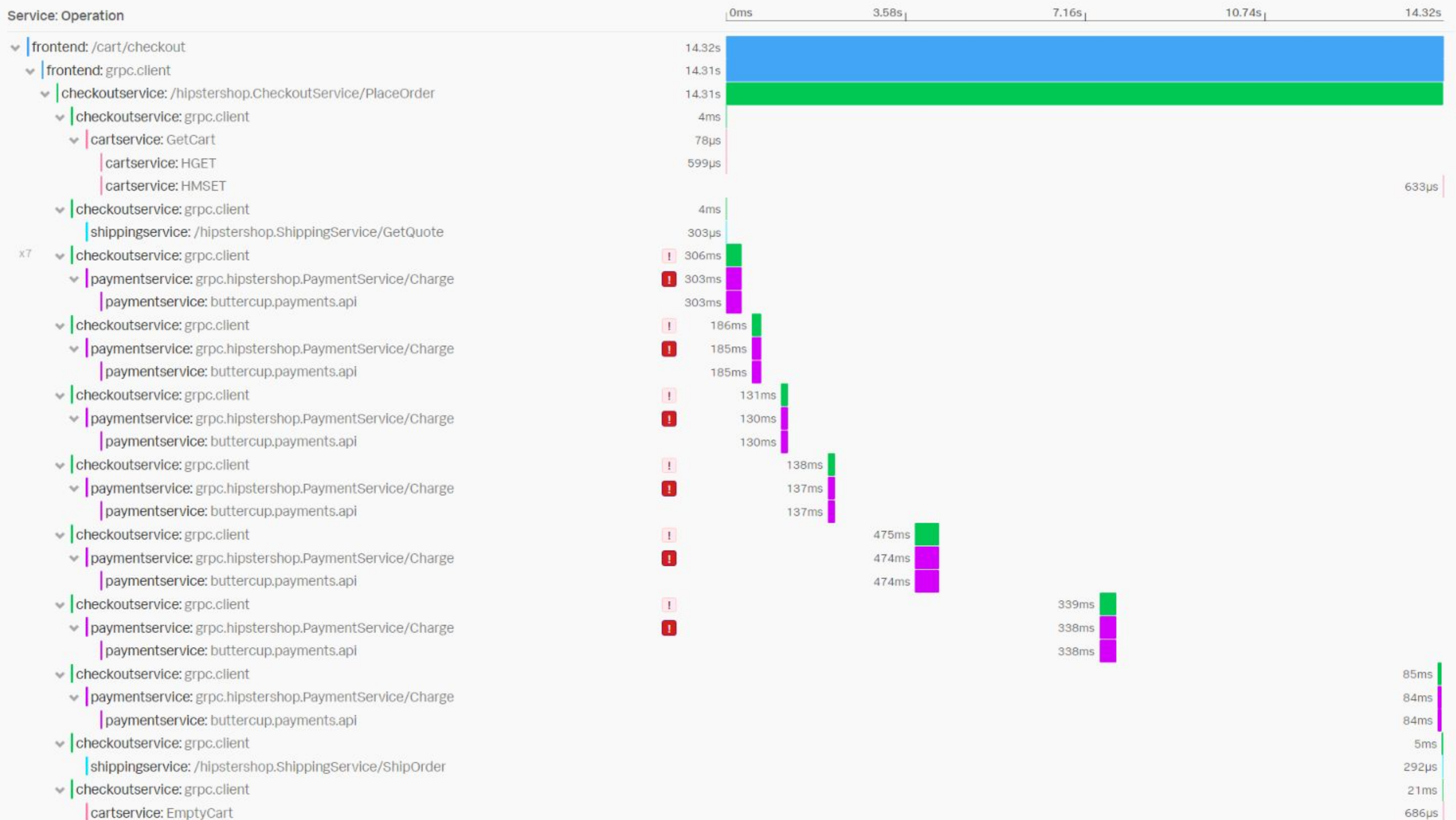
Example: Errors Causing Latency



Example: Errors Causing Latency



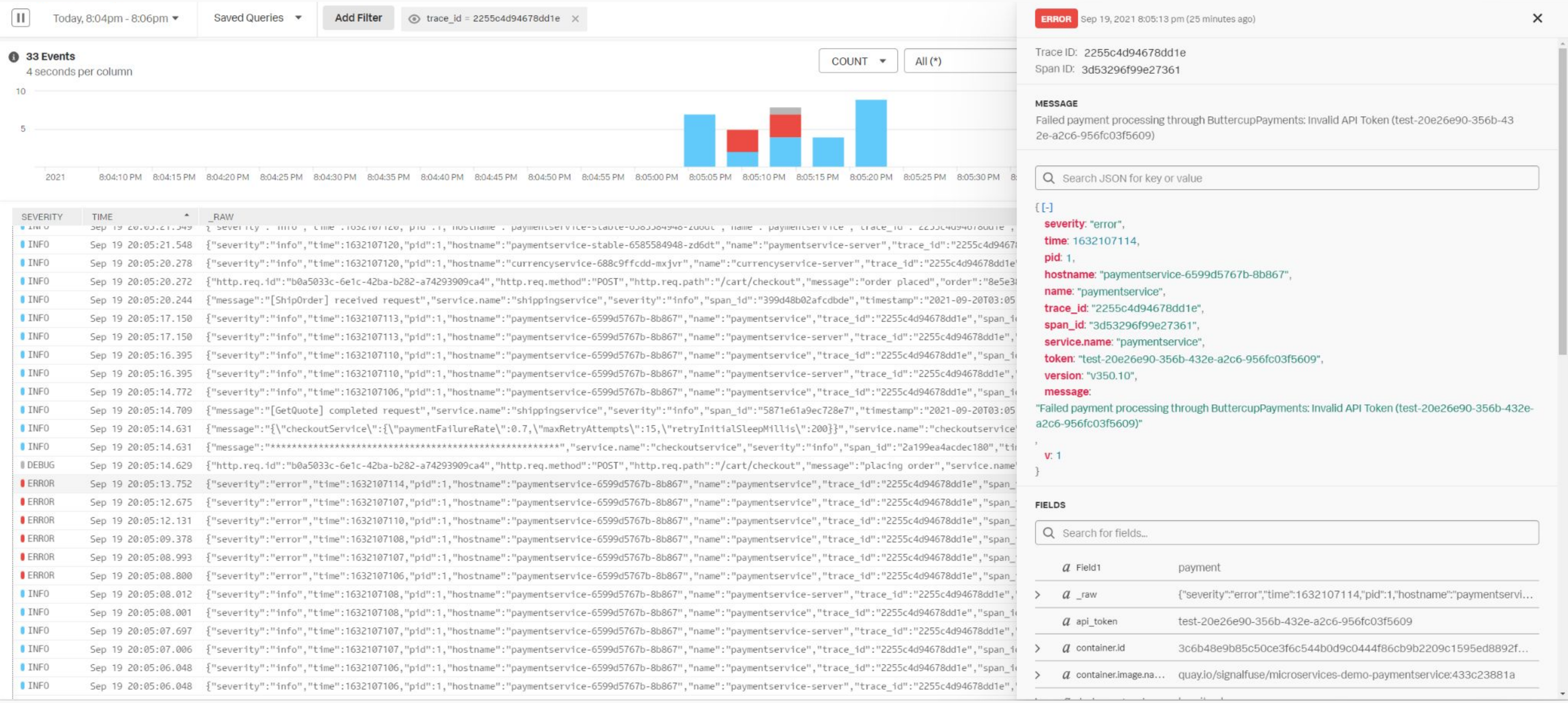
Example: Errors Causing Latency



```
Service: Operation                                                      0ms    3.58s    7.16s    10.74s    14.32s

frontend: /cart/checkout                                                    14.32s
  frontend: grpc.client                                                     14.31s
    checkoutservice: /hipstershop.CheckoutService/PlaceOrder                14.31s
      checkoutservice: grpc.client                                          4ms
        cartservice: GetCart                                                78µs
          cartservice: HGET                                                 599µs
          cartservice: HMSET                                                633µs
      checkoutservice: grpc.client                                          4ms
        shippingservice: /hipstershop.ShippingService/GetQuote              303µs
x7    checkoutservice: grpc.client                                      !   306ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   303ms
          paymentservice: buttercup.payments.api                            303ms
      checkoutservice: grpc.client                                      !   186ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   185ms
          paymentservice: buttercup.payments.api                            185ms
      checkoutservice: grpc.client                                      !   131ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   130ms
          paymentservice: buttercup.payments.api                            130ms
      checkoutservice: grpc.client                                      !   138ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   137ms
          paymentservice: buttercup.payments.api                            137ms
      checkoutservice: grpc.client                                      !   475ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   474ms
          paymentservice: buttercup.payments.api                            474ms
      checkoutservice: grpc.client                                      !   339ms
        paymentservice: grpc.hipstershop.PaymentService/Charge          !   338ms
          paymentservice: buttercup.payments.api                            338ms
      checkoutservice: grpc.client                                          85ms
        paymentservice: grpc.hipstershop.PaymentService/Charge              84ms
          paymentservice: buttercup.payments.api                            84ms
      checkoutservice: grpc.client                                          5ms
        shippingservice: /hipstershop.ShippingService/ShipOrder             292µs
      checkoutservice: grpc.client                                          21ms
        cartservice: EmptyCart                                              686µs
```

Example: Errors Causing Latency



```
SEVERITY   TIME                  _RAW
INFO       Sep 19 20:05:21.549   {"severity":"info","time":1632107120,"pid":1,"hostname":"paymentservice-stable-6585584948-zd6dt","name":"paymentservice","trace_id":"2255c4d94678dd1e",
INFO       Sep 19 20:05:21.548   {"severity":"info","time":1632107120,"pid":1,"hostname":"paymentservice-stable-6585584948-zd6dt","name":"paymentservice-server","trace_id":"2255c4d9467
INFO       Sep 19 20:05:20.278   {"severity":"info","time":1632107120,"pid":1,"hostname":"currencyservice-688c9ffcdd-mxjvr","name":"currencyservice-server","trace_id":"2255c4d94678dd1e"
INFO       Sep 19 20:05:20.272   {"http.req.id":"b0a5033c-6e1c-42ba-b282-a74293909ca4","http.req.method":"POST","http.req.path":"/cart/checkout","message":"order placed","order":"8e5e3
INFO       Sep 19 20:05:20.244   {"message":"[ShipOrder] received request","service.name":"shippingservice","severity":"info","span_id":"399d48b02afcdbde","timestamp":"2021-09-20T03:05
INFO       Sep 19 20:05:17.150   {"severity":"info","time":1632107113,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:17.150   {"severity":"info","time":1632107113,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice-server","trace_id":"2255c4d94678dd1e","
INFO       Sep 19 20:05:16.395   {"severity":"info","time":1632107110,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:16.395   {"severity":"info","time":1632107110,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice-server","trace_id":"2255c4d94678dd1e","
INFO       Sep 19 20:05:14.772   {"severity":"info","time":1632107106,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:14.709   {"message":"[GetQuote] completed request","service.name":"shippingservice","severity":"info","span_id":"5871e61a9ec728e7","timestamp":"2021-09-20T03:05
INFO       Sep 19 20:05:14.631   {"message":"{\"checkoutService\":{\"paymentFailureRate\":0.7,\"maxRetryAttempts\":15,\"retryInitialSleepMillis\":200}}","service.name":"checkoutservice"
INFO       Sep 19 20:05:14.631   {"message":"*******************************************************","service.name":"checkoutservice","severity":"info","span_id":"2a199ea4acdec180","ti
DEBUG      Sep 19 20:05:14.629   {"http.req.id":"b0a5033c-6e1c-42ba-b282-a74293909ca4","http.req.method":"POST","http.req.path":"/cart/checkout","message":"placing order","service.name"
ERROR      Sep 19 20:05:13.752   {"severity":"error","time":1632107114,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
ERROR      Sep 19 20:05:12.675   {"severity":"error","time":1632107107,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
ERROR      Sep 19 20:05:12.131   {"severity":"error","time":1632107110,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
ERROR      Sep 19 20:05:09.378   {"severity":"error","time":1632107108,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
ERROR      Sep 19 20:05:08.993   {"severity":"error","time":1632107107,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
ERROR      Sep 19 20:05:08.800   {"severity":"error","time":1632107106,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_
INFO       Sep 19 20:05:08.012   {"severity":"info","time":1632107108,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice-server","trace_id":"2255c4d94678dd1e","
INFO       Sep 19 20:05:08.001   {"severity":"info","time":1632107108,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:07.697   {"severity":"info","time":1632107107,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice-server","trace_id":"2255c4d94678dd1e","
INFO       Sep 19 20:05:07.006   {"severity":"info","time":1632107107,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:06.048   {"severity":"info","time":1632107106,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice","trace_id":"2255c4d94678dd1e","span_i
INFO       Sep 19 20:05:06.048   {"severity":"info","time":1632107106,"pid":1,"hostname":"paymentservice-6599d5767b-8b867","name":"paymentservice-server","trace_id":"2255c4d94678dd1e","
```

ERROR Sep 19, 2021 8:05:13 pm (25 minutes ago)

Trace ID: 2255c4d94678dd1e
Span ID: 3d53296f99e27361

MESSAGE
Failed payment processing through ButtercupPayments: Invalid API Token (test-20e26e90-356b-432e-a2c6-956fc03f5609)

Search JSON for key or value

```
{[-]
  severity: "error",
  time: 1632107114,
  pid: 1,
  hostname: "paymentservice-6599d5767b-8b867",
  name: "paymentservice",
  trace_id: "2255c4d94678dd1e",
  span_id: "3d53296f99e27361",
  service.name: "paymentservice",
  token: "test-20e26e90-356b-432e-a2c6-956fc03f5609",
  version: "v350.10",
  message:
"Failed payment processing through ButtercupPayments: Invalid API Token (test-20e26e90-356b-432e-a2c6-956fc03f5609)"
,
  v: 1
}
```

FIELDS

Search for fields...

Field1	payment
_raw	{"severity":"error","time":1632107114,"pid":1,"hostname":"paymentservi…
api_token	test-20e26e90-356b-432e-a2c6-956fc03f5609
container.id	3c6b48e9b85c50ce3f6c544b0d9c0444f86cb9b2209c1595ed8892f…
container.image.na…	quay.io/signalfuse/microservices-demo-paymentservice:433c23881a

Example: Errors Causing Latency

How did we do this?

- Correlate application metrics with services
- Correlate host metrics with services
- Propagate request data all the way down the stack, correlate it with metrics
- Correlate logs with traces
- Correlate logs with services

Example: Errors Causing Latency

Ok, but how did we actually do this?

- Use OpenTelemetry SDKs or language instrumentation to capture traces and application metrics
  - Send to the Collector
- Use the OpenTelemetry Collector to capture host metrics, host logs, and application logs

Important configuration

- Set service name in resource metadata
- Stamp trace and span IDs in logs

Example: Crashes Causing Cost

Service is causing latency but there is nothing new in the traces or logs.

- Elevated CPU usage from nodes
- Breaking down CPU usage reveals simulations are impacted
- Searching for traces and logs reveals nothing new
- Simulation engine RPCs experience slight new latency

Example: Crashes Causing Cost

Metrics, traces or logs are not useful. What else is available?

- Crash dumps from the nodes reveal consistent crashes
- Crash occurs after serving the RPC
- Each incoming request causes a cold start
- Debugging code locally reveals service panics

Simulation engine was crashing after handling each request and new one was being forked. Although it caused little new latency, it elevated the CPU usage and impacted efficiency.

Example: Library Causing Latency

Service begins to experience latency and elevated CPU usage.

- A large number of changes have been pushed recently
- Checking the cluster reveals no impact to other services
- Traces show nothing new, no retries or outbound request issues

Example: Library Causing Latency

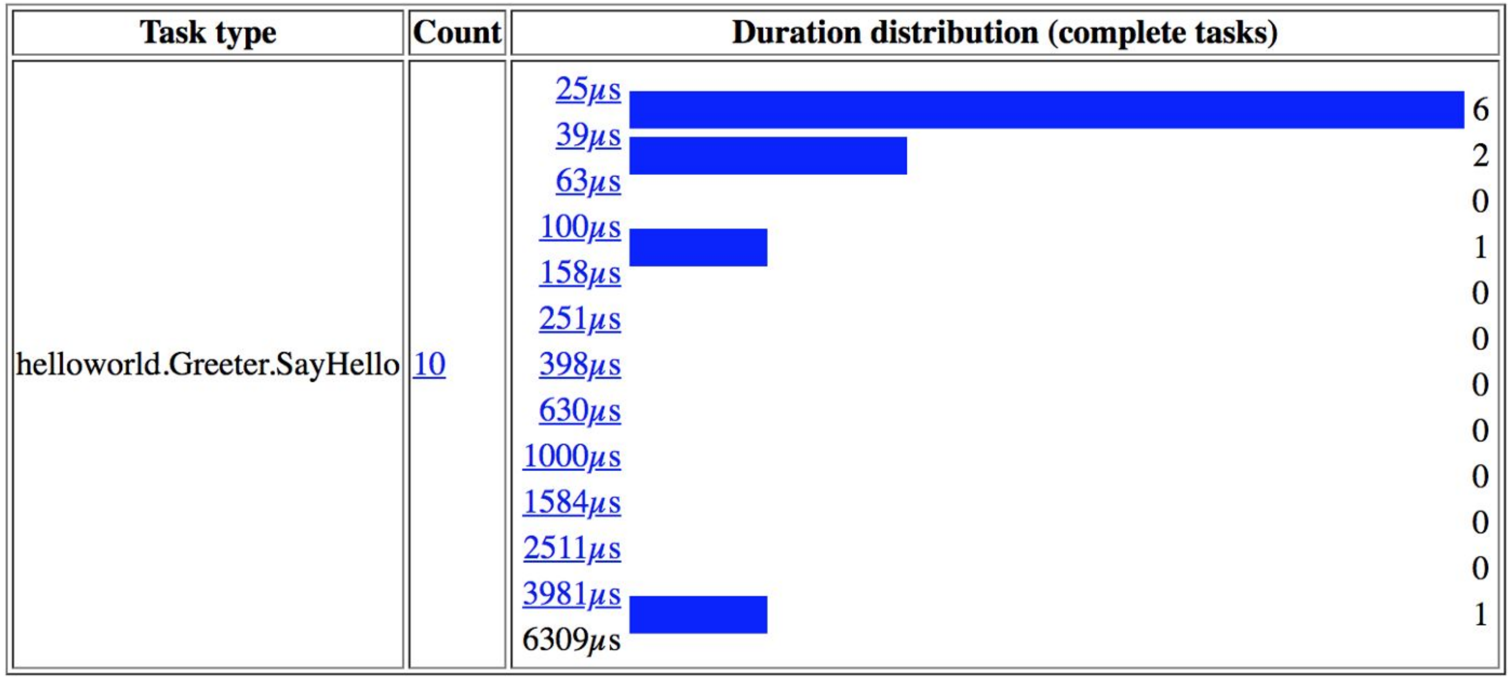
Reviewing diffs reveals new versions of library dependencies.

- Briefly enabling profiling confirms
- Profiles reveal only a certain RPC is impacted
- Reading RPC handler reveals the use a new compression library

Service was 50% more CPU intensive due to the new compression library. Rolling back mitigated the issue rather than running out of CPU capacity.

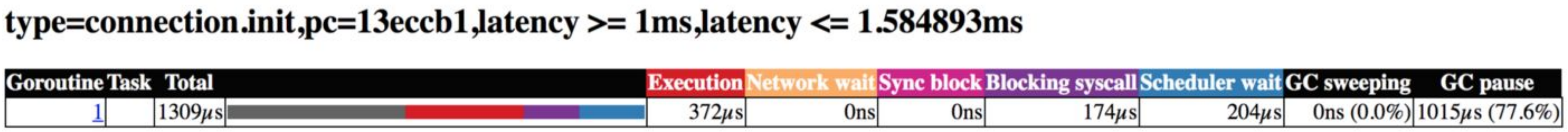
Example: Library Causing Latency

Since 1.11, Go runtime tracer can report events in the lifetime of an RPC.



Task type	Count	Duration distribution (complete tasks)
helloworld.Greeter.SayHello	10	

25µs	6
39µs	2
63µs	0
100µs	1
158µs	0
251µs	0
398µs	0
630µs	0
1000µs	0
1584µs	0
2511µs	0
3981µs	1
6309µs	



type=connection.init,pc=13eccb1,latency >= 1ms,latency <= 1.584893ms

Goroutine	Task	Total		Execution	Network wait	Sync block	Blocking syscall	Scheduler wait	GC sweeping	GC pause
1		1309µs		372µs	0ns	0ns	174µs	204µs	0ns (0.0%)	1015µs (77.6%)

Correlations

Types of correlations:

- Correlations among metrics, logs, traces
- Resource inferences
- Local and propagated annotations

Correlations don’t just make it easier to navigate telemetry but also make it easier to rule out non-issues quickly.

What's Next?

- Probably profiles
    - Will take a while; need to ship metrics and finish logs first)
    - Can correlate performance data all the way down to code!
- Core dumps, crashes, exception call stacks
- Associations with resource information gained from eBPF
    - Correlate network performance information with traces!
- Correlations with language runtime traces

RESILIENCE

REALIZED

KubeCon | CloudNativeCon

North America 2021