Strategic Sampling: Architectural Approaches to Efficient Telemetry

Fosdem 2024

About

Benedikt Bongartz
Senior Software Engineer
Red Hat

frzifus

frzifus:matrix.org

Julius Hinze
Software Engineer
Cisco

juliusmh

Agenda

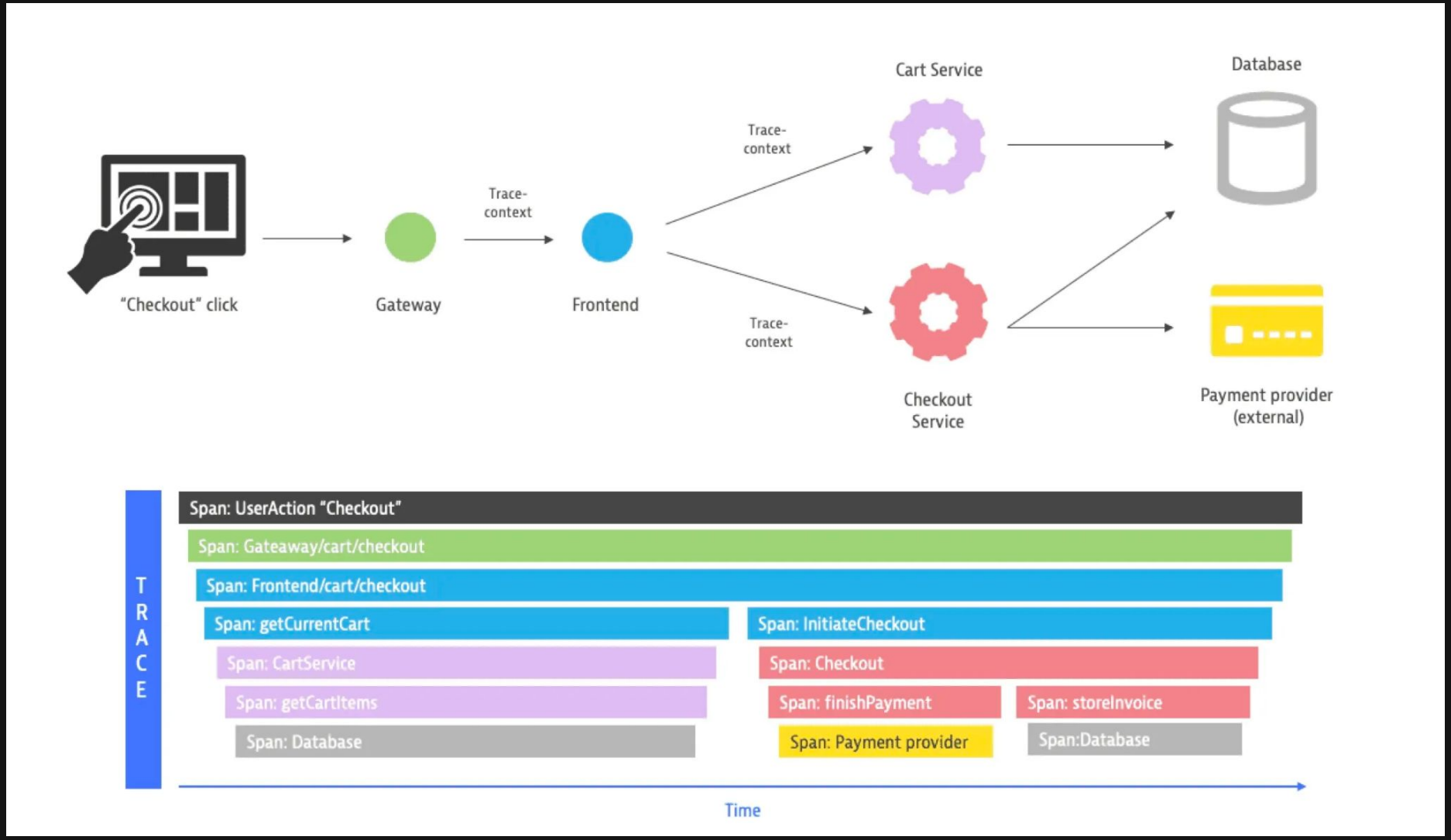
- OpenTelemetry in a Nutshell
- Sampling in OpenTelemetry
  - What does it mean and why is it important?
  - Comparing Sampling Approaches
- Challenges of Sampling
  - Connection handling
  - How to survive unpredictable load? - (Auto) scaling
- Conclusion
- Q&A Session

OpenTelemetry in a Nutshell

OpenTelemetry

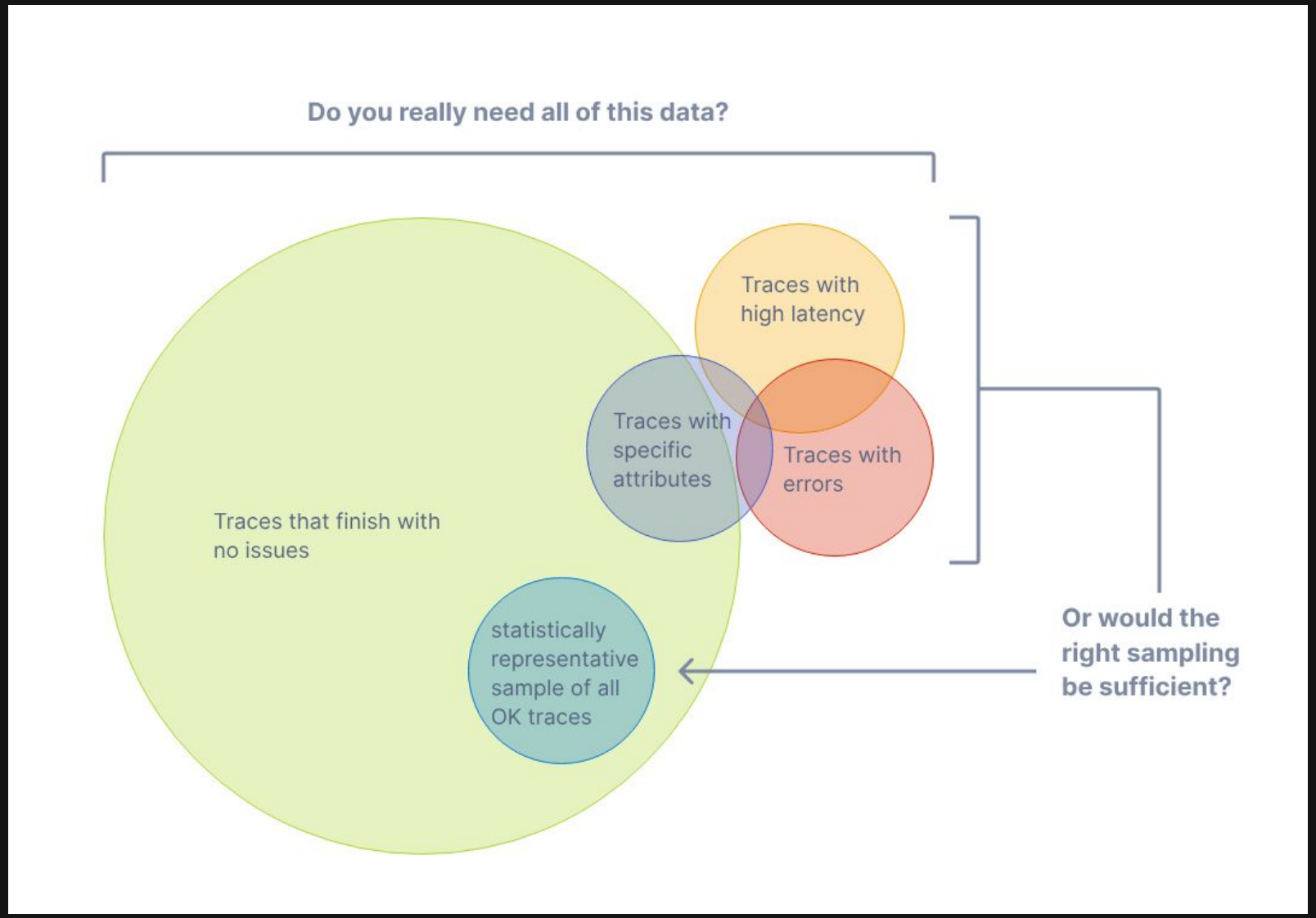
- Open source!
- Cloud Native Computing Foundation (CNCF)
- Vendor neutral telemetry data collection
- Telemetry = Traces + Metrics + Logs
- Specification, API, SDK, data model - OTLP, auto-instrumentation, collector
- Helm chart, Kubernetes operator

Introduction Tracing



https://henesgokdag.medium.com/distributed-tracing-9300d55e7245

Sampling



https://opentelemetry.io/docs/concepts/sampling/

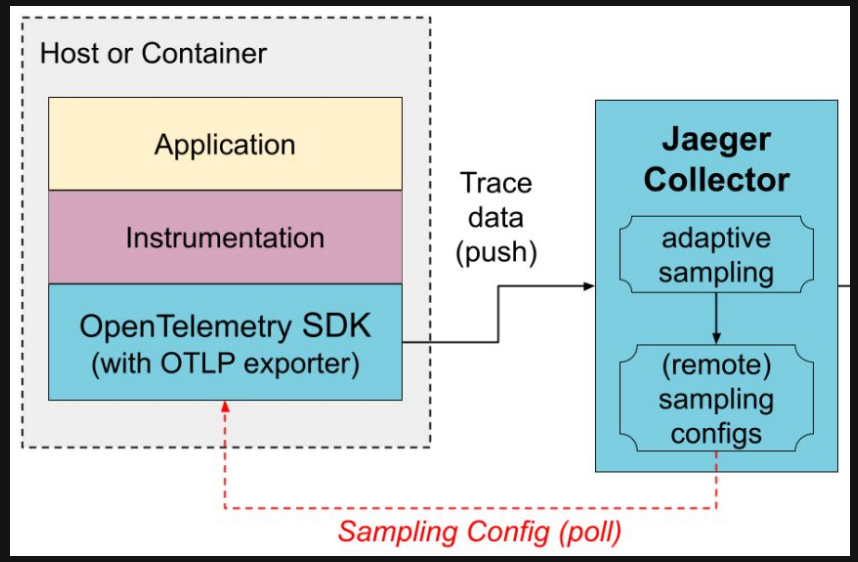
Example: AWS XRay cost

- Setup (production environment, telco workloads)
  - ~30 Microservices, ~110 Nodes, ~2350 Pods, ~940 CPU
- Calculated cost with **100%** sampling
  - 1.100.000 traces/min → 1.100.000 * 60 * 24 * 30 * 100% = 47.500.200.000
  - = 237.600 $ (27.01.2024, region=eu-west-1)
- Calculated cost with **0.1%** sampling
  - 1.100.000 traces/min → 1.100.000 * 60 * 24 * 30 * 0.1% = 4.752.000.000
  - = 237 $ (27.01.2024, region=eu-west-1)
- How can we choose the 0.1% ?

https://calculator.aws/#/createCalculator/xray

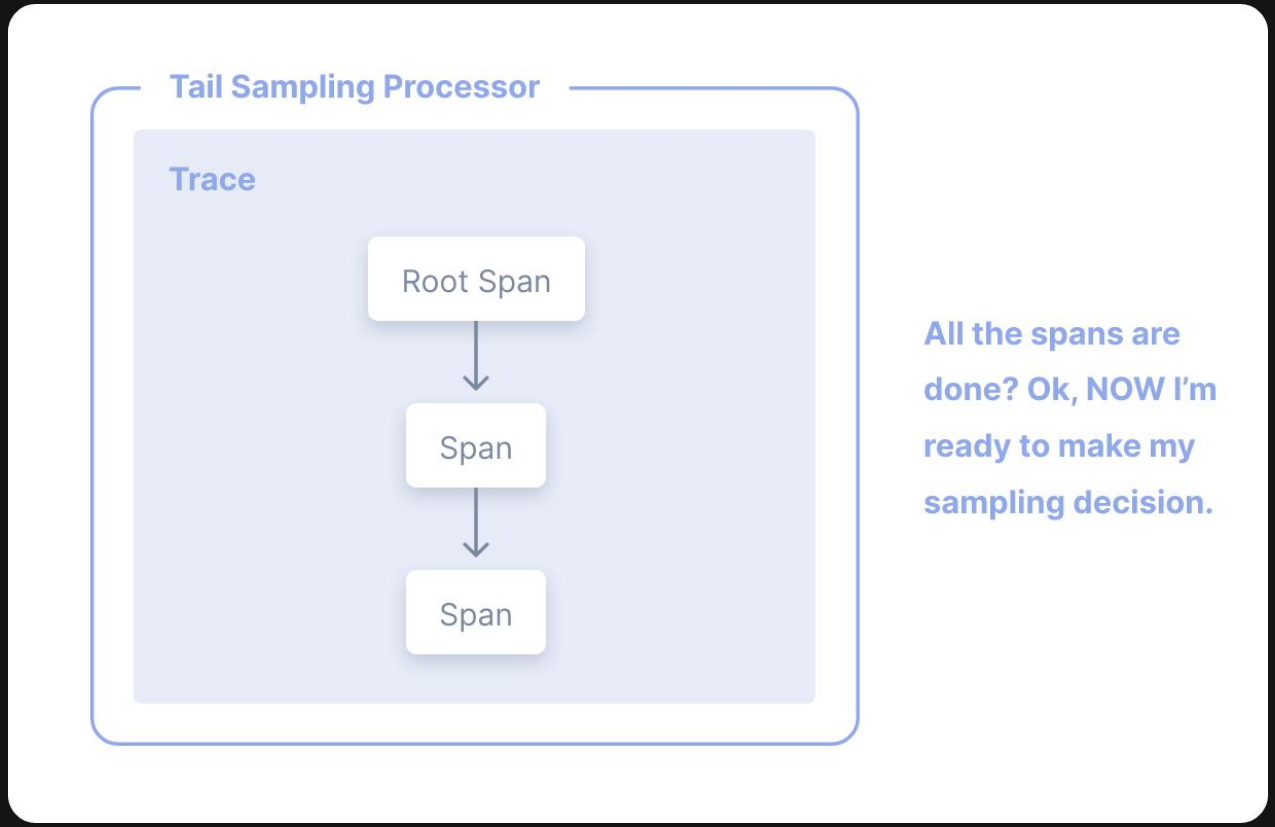
Head-based sampling

- Sampling decision is made at the **beginning** of a trace
- Efficient, Easy to understand and to configure
- Common available options (more):
  - Parent-based
  - Probability
- SDK needs to be configured
  - Manually e.g. via environment variables
  - Jaeger Remote Sampling extension (docs)
- Alternative Probabilistic sampler processor (docs)



Tail-based sampling

- Sampling decision is made at the **end** of a trace
- The decision maker needs to be aware of all spans of a trace
- Allows complex policies
- **Consumes extra resources**
- Use tail-based sampling when you want to investigate *rare* or *extreme cases* that might have significant impact or need special attention



https://opentelemetry.io/docs/concepts/sampling#tail-sampling

Tail-based sampling

- Calculated cost telo/xray setup - 1.1 Mio traces per Minute
  - **100%** sampling => 237.600 $
  - **0.1%** sampling => 237 $ + (sampling cost)
- Sampling resource cost
  - (Cluster resource limits 15000Mi + 6500m CPU)
  - Single Instance example
    - Type: t2.xlarge per H: 0,1856 USD Cores: 4 Mem: 16 GiB
      - Xray(237 $) + t2.xlarge(**133 $**) = **370 $**
    - Type: t2.2xlarge per H: 0,3712 USD Cores: 8 Mem: 32 GiB
      - Xray(237 $) + t2.2xlarge(**267 $**) = **504 $**

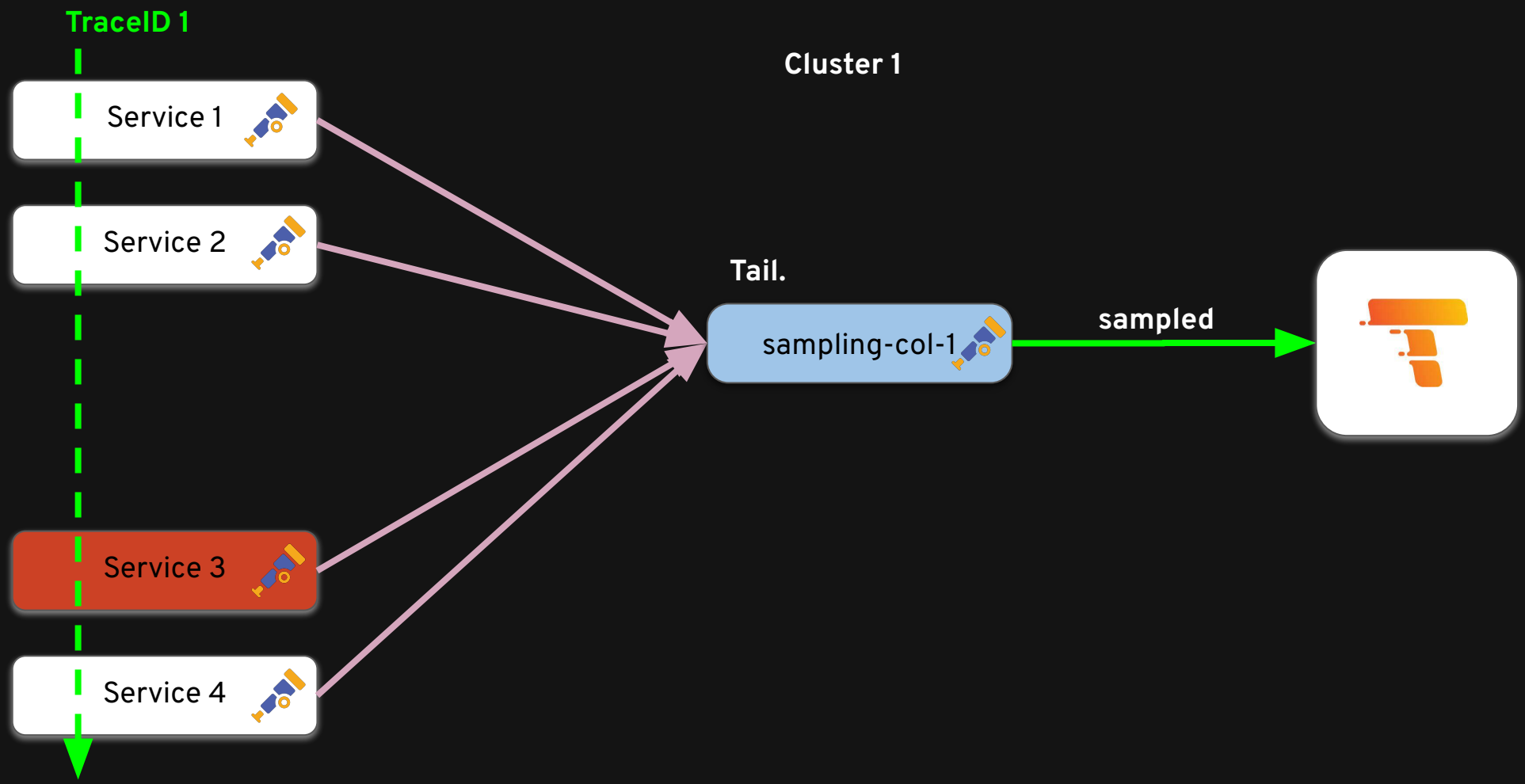
CISCO

Red Hat

How to apply Tail-based sampling

using the OpenTelemetry Collector

Tail-based sampling



Tail-based sampling

```yaml
apiVersion: opentelemetry.io/v1alpha1
kind: OpenTelemetryCollector
metadata:
 name: sampling-col
spec:
 mode: deployment
 replicas: 5
 resources: ...
 config: |
   receivers:
     otlp:
   processors:
     memory_limiter: ...
     batch/traces: ...
     tail_sampling: ...
   exporters:
     otlp/tempo: ...
   service:
     pipelines:
       traces:
         receivers: [otlp]
         processors: [memory_limiter, batch/traces, tail_sampling]
         exporters: [otlp/tempo]
```

```yaml
resources:
  requests:
    memory: "3000Mi"
    cpu: "1300m"
  limits:
    memory: "3000Mi"
    cpu: "1300m"
```

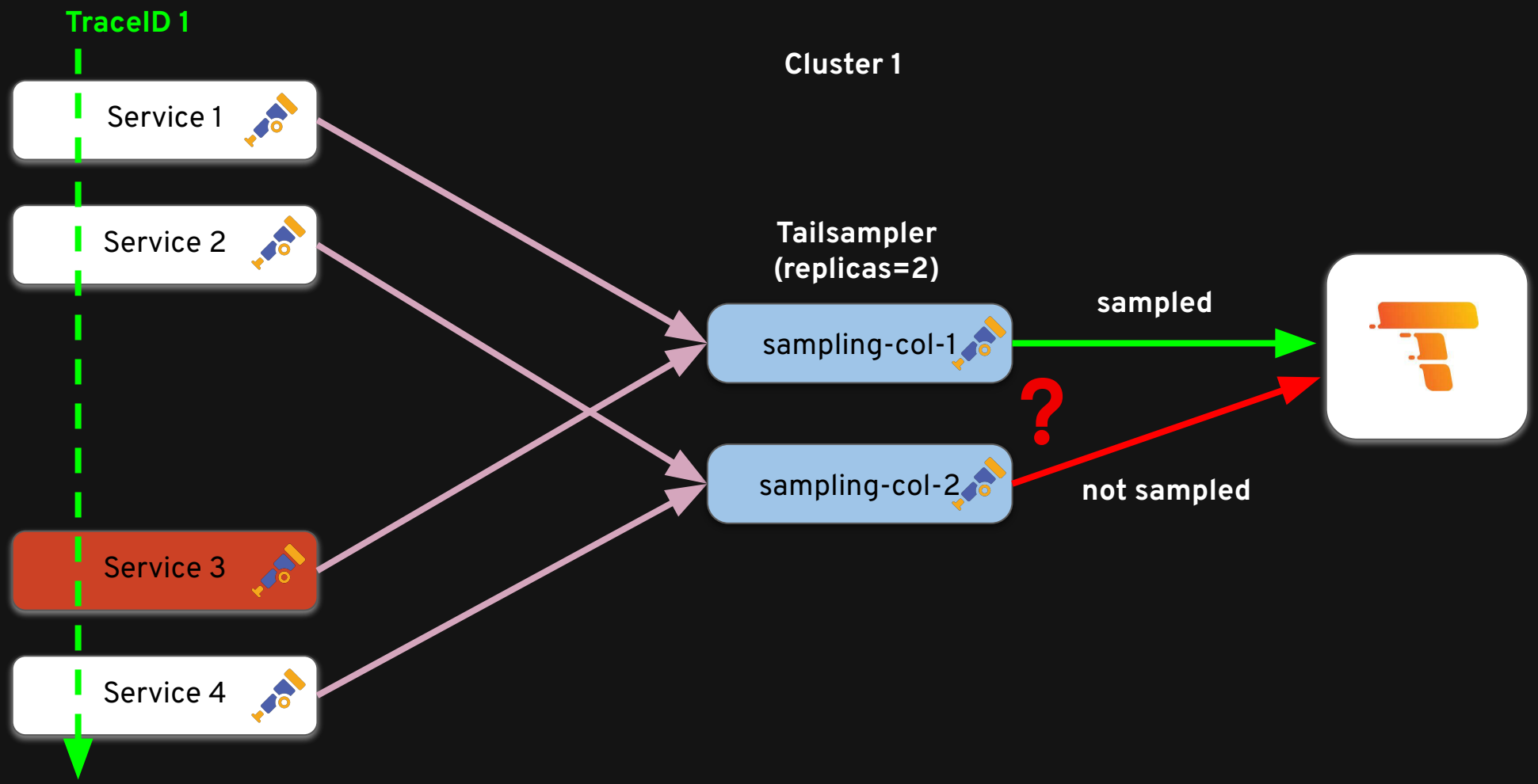
```yaml
tail_sampling:
  decision_wait: 10s
  num_traces: 100000
  # expected_new_traces_per_sec: 5000
  policies:
    - name: policy-errors-retain
      type: status_code
      status_code: {status_codes: [ERROR]}
    - name: policy-probabilistic
      type: probabilistic
      probabilistic:
        sampling_percentage: 10
```

Multiple policies exist today: (docs)

CISCO

Red Hat

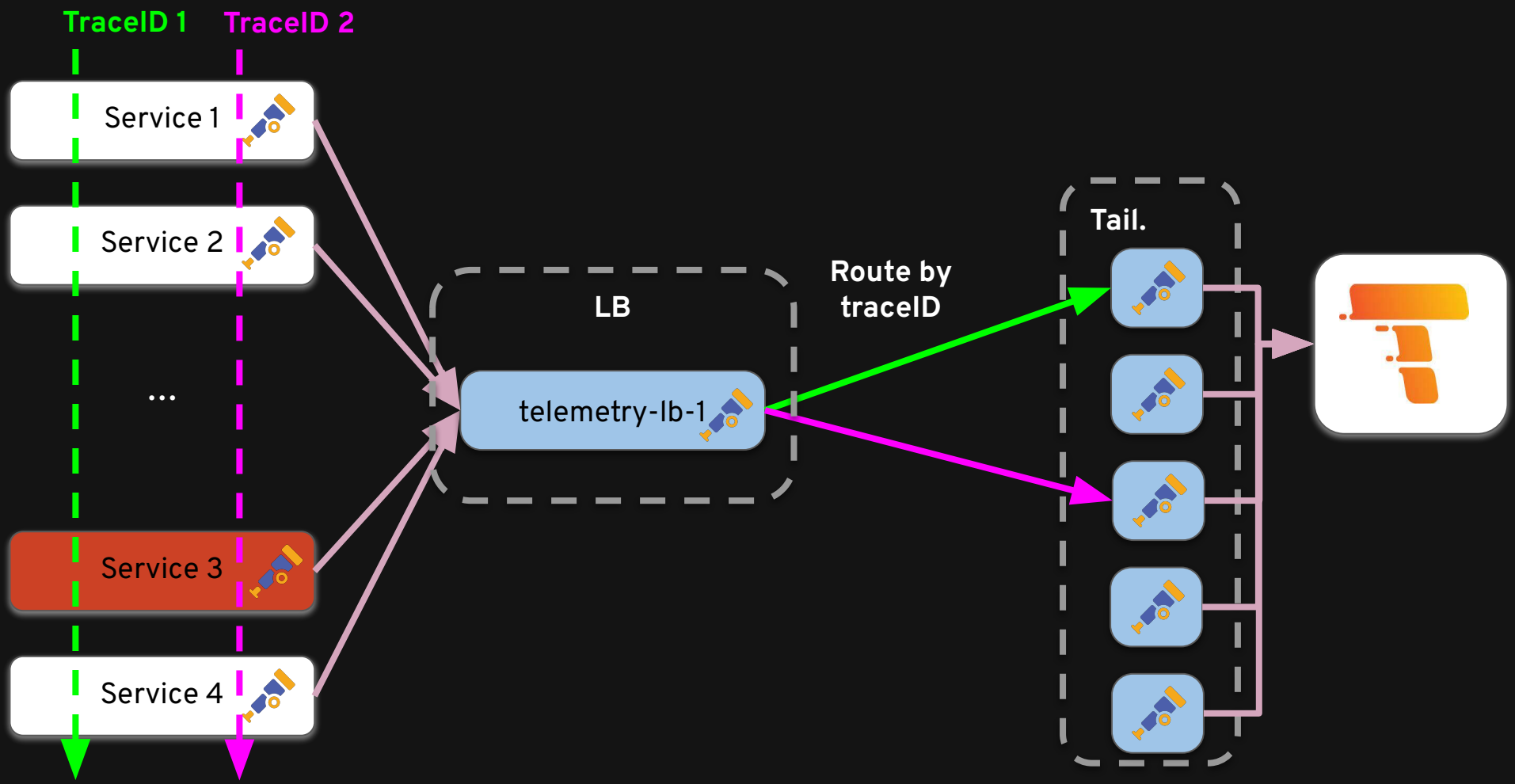
Tail-based sampling



Scaling out

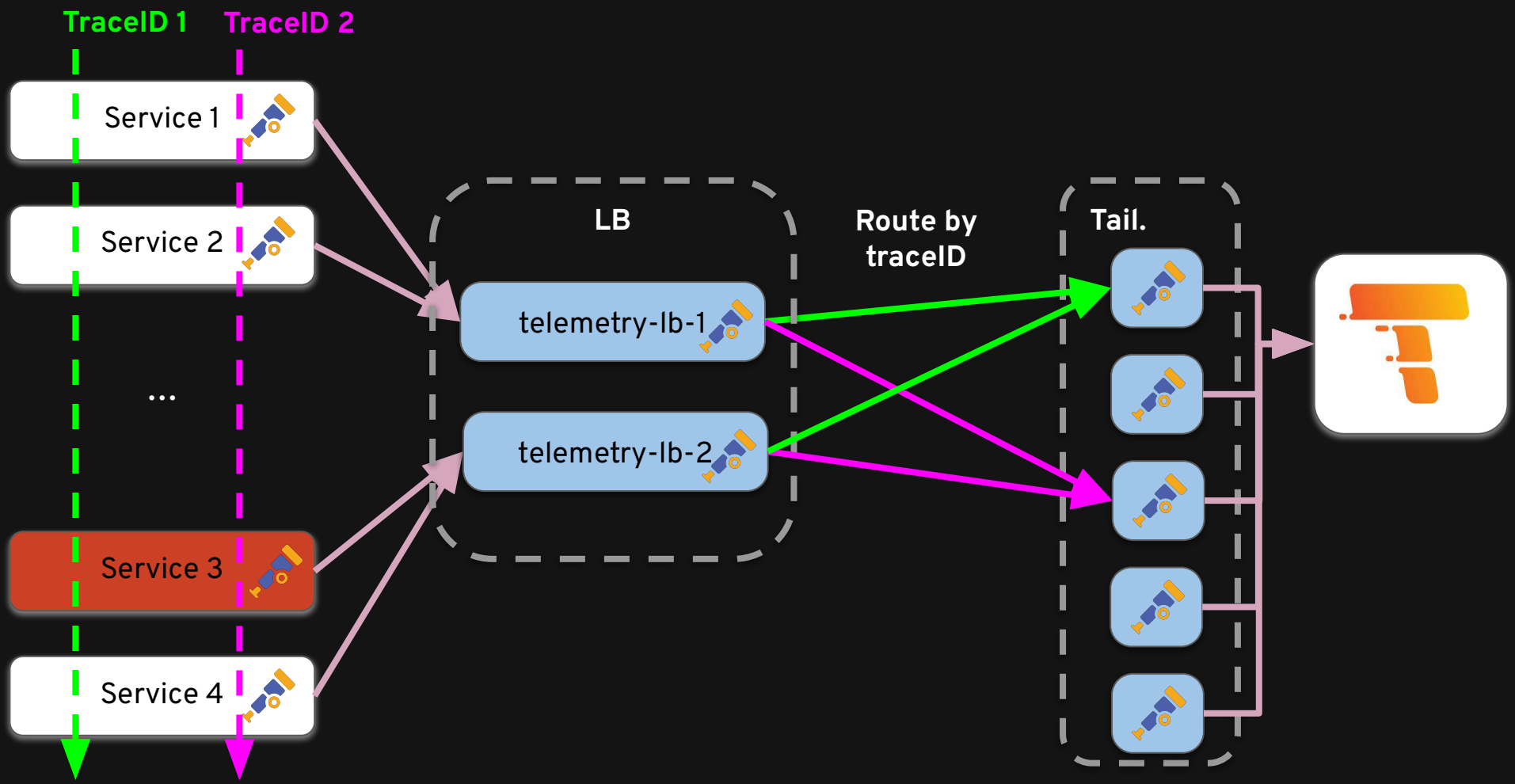
Layered collectors

Tail-based sampling: load balancing



Optionally, we can generate RED metrics before dropping traces.

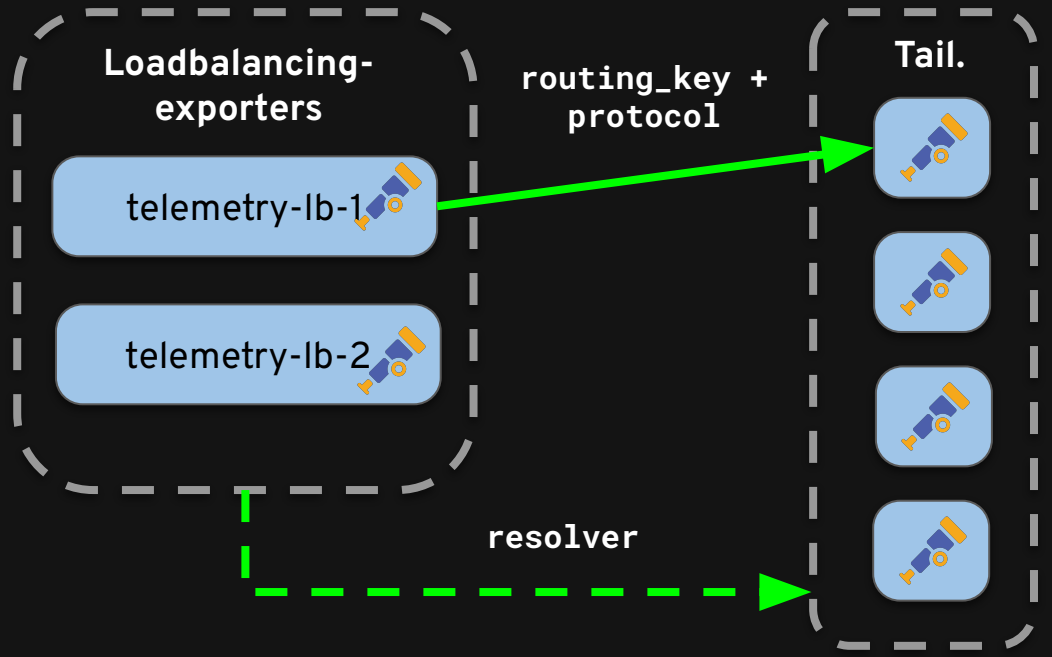
Tail-based sampling: load balancing



Optionally, we can generate RED metrics before dropping traces.

Load-balancing exporter

- Resolver
  - Find upstream collectors
  - Supported: DNS, k8s service, static backends
- Protocol
  - used to send traces/metrics/logs upstream
- Routing key + consistent hash ring



Load-balancing exporter

```
apiVersion: opentelemetry.io/v1alpha1
kind: OpenTelemetryCollector
metadata:
 name: telemetry-lb
spec:
 mode: deployment
 replicas: 2
 config: |
   exporters:
     loadbalancing:
       routingKey: traceID
       protocol:
         otlp:
           sending_queue:
             queue_size: 4000
       resolver:
         dns:
           hostname: telemetry-collector-headless.telemetry.svc.cluster.local
           port: 4317
   service:
     pipelines:
       traces:
         exporters: [loadbalancing]
```

```
resolver:
     k8s:
       service: lb-svc.kube-public
       ports:
         - 15317
         - 16317
resolver:
  static:
    hostnames:
    - backend-1:4317
    - backend-2:4317
    - backend-3:4317
    - backend-4:4317
```

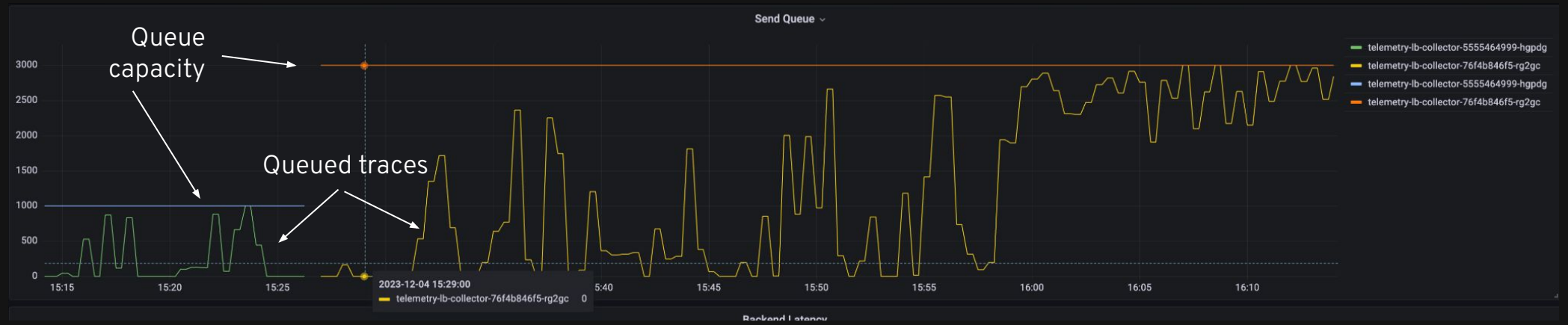
Problem solved?

Challenges

- LBs only work if **exporting** is faster than **receiving**
- Very sensitive to upstream problems
- But, easy to spot and debug:
  - `otelcol_exporter_queue_capacity`
  - `otelcol_exporter_queue_size`
  - `otelcol_loadbalancer_backend_latency`

https://opentelemetry.io/docs/collector/scaling/#when-to-scale

Queue Size?



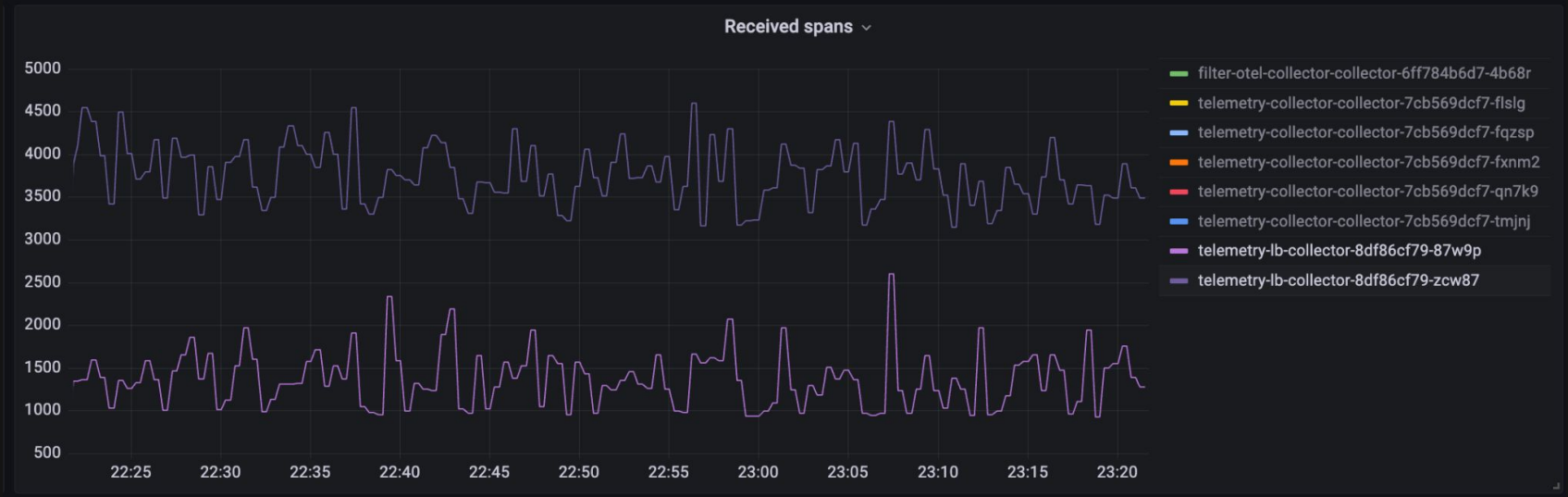
https://opentelemetry.io/docs/collector/scaling/

CISCO

Red Hat

LB inception

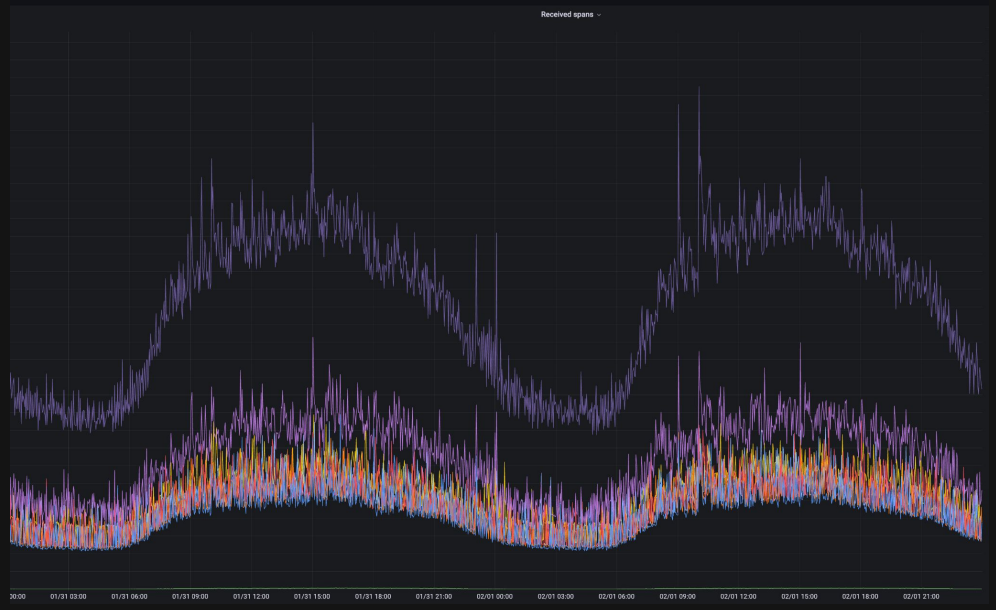
- How to lb the LBs?
- K8s LB (L4)? doesn't handle gRPC well
- use otlphttp instead? Less efficient
- use L7 lb, e.g. envoy?
- Deploy in sidecar mode?



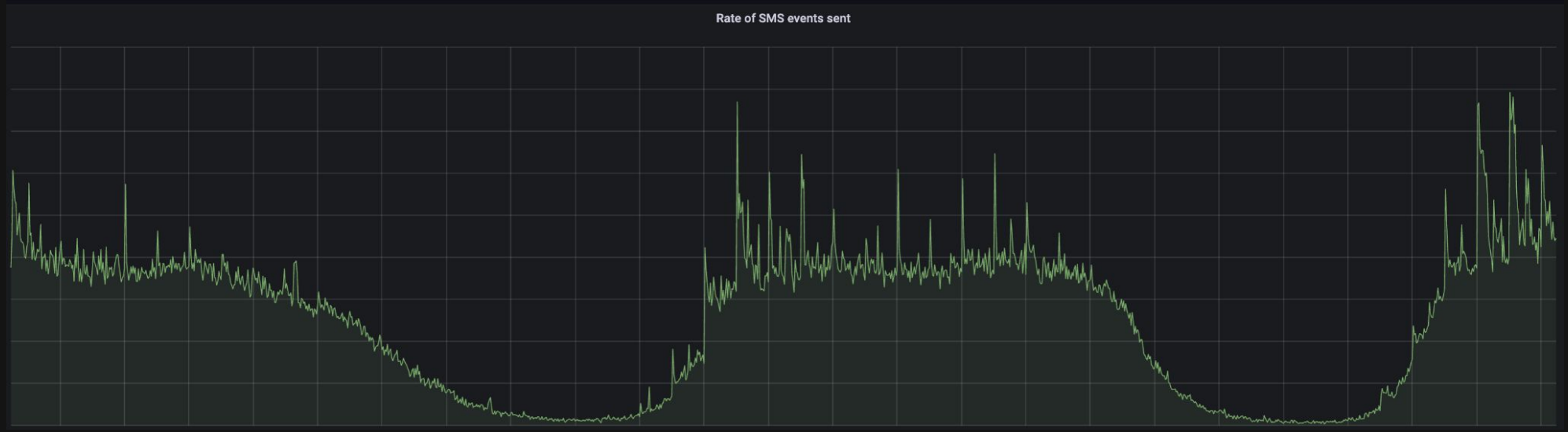
Auto-scaling?

- Doesn't exist (yet)
- Save resources when traffic is low
- Resolvers not "termination aware"
- Errors appear during bursts

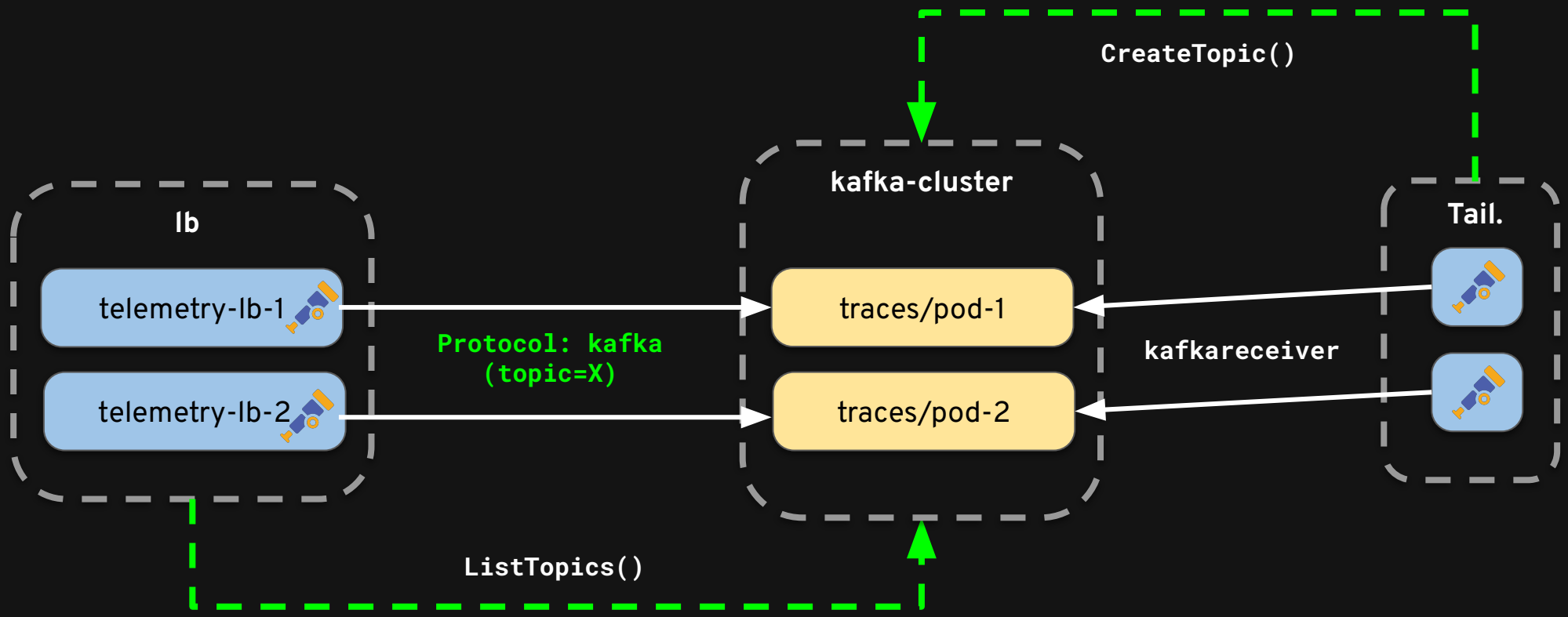
Received spans



Rate of SMS sent



Auto-scaling? ideas...



Auto-scaling? ideas...

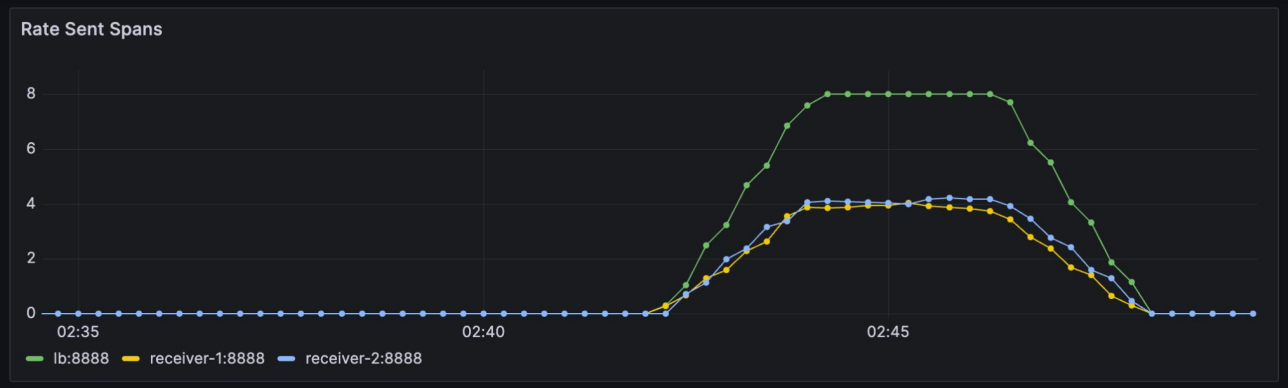
- Simple PoC in ~500 LOC (most of it copy pasta)
- Kafkaresolver
  - ListTopics with prefix, every n seconds
- Protocol: kafka
  - Recycle kafkaexporter factory
- Kafkareceiver
  - Create topic on Start()

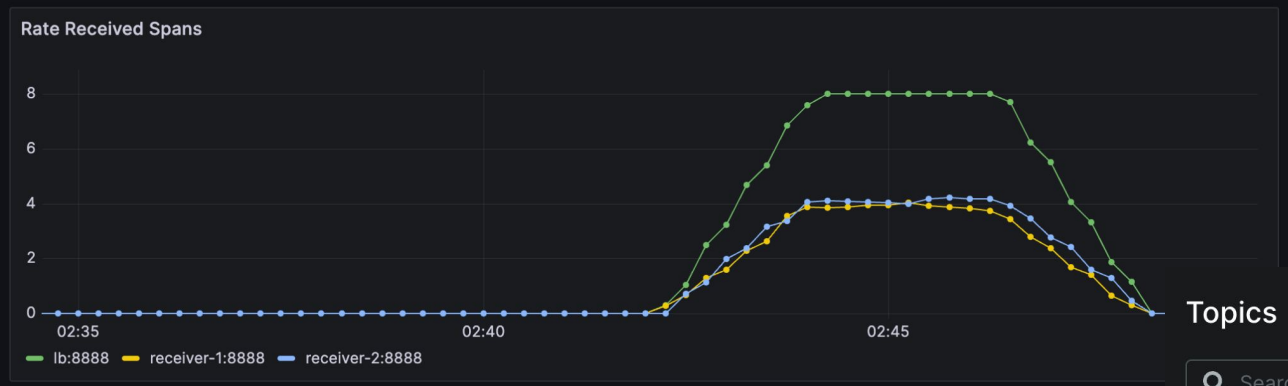
```
resolver:
  kafka:
    brokers:
      - kafka:9092
    protocol_version: 2.0.0
    timeout: 5s
    topic_prefix: otel-pod-
```

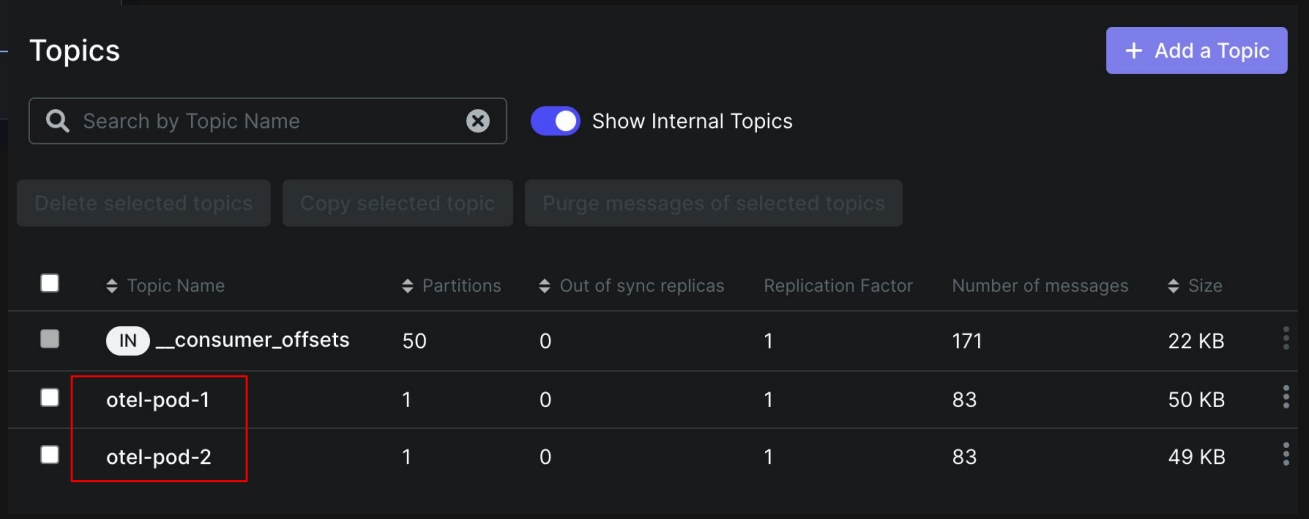
```
protocol:
  kafka:
    brokers:
      - kafka:9092
    protocol_version: 2.0.0
    encoding: otlp_proto
```

```
receivers:
  kafka:
    brokers:
      - kafka:9092
    metadata:
      full: true
    protocol_version: 2.0.0
    topic: otel-pod-1
    create_topic: true
    encoding: otlp_proto
```

Auto-scaling? ideas...







Conclusion

Conclusion

- Traces are valuable for understanding system behavior
  - But storing all traces is costly
- Head/tail sampling can reduce trace volume and focus on important data
  - Cost is a significant factor in trace management
- Tail-based sampling configurations can be complex
- Load balancing can help manage high traffic loads in trace systems
- Easy to implement customized solutions on top of OTEL

Thank you

Benedikt Bongartz
@frzifus

Julius Hinze
@juliusmh