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

- 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

Do you really need all of this data?

Traces that finish with no issues

- Traces with high latency
- Traces with specific attributes
- Traces with errors

Or would the right sampling be sufficient?

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 $
How to apply Tail-based sampling

using the OpenTelemetry Collector
Tail-based sampling

Cluster 1

Service 1

Service 2

Service 3

Service 4

Tail.

sampled

TraceID 1

sampling-col-1
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]
resources:
  requests:
    memory: "3000Mi"
    cpu: "1300m"
  limits:
    memory: "3000Mi"
    cpu: "1300m"

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)
```
Tail-based sampling

Cluster 1

Service 1

Service 2

Service 3

Service 4

Tailsampler (replicas=2)
sampling-col-1

sampling-col-2

TraceID 1

Cluster 1

sampled

not sampled
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

```yaml
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
    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

service:
  pipelines:
    traces:
      exporters: [loadbalancing]
```
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/
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
Auto-scaling? ideas...

```
<table>
<thead>
<tr>
<th><code>Protocol: kafka (topic=X)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>telemetry-lb-1</td>
</tr>
<tr>
<td>telemetry-lb-2</td>
</tr>
<tr>
<td>kafka-cluster</td>
</tr>
<tr>
<td>traces/pod-1</td>
</tr>
<tr>
<td>traces/pod-2</td>
</tr>
<tr>
<td>kafkareceiver</td>
</tr>
</tbody>
</table>

ListTopics() -> kafka-cluster
CreateTopic() -> kafka-cluster
Tail. -> kafkareceiver

25
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
    protocol_version: 2.0.0
    encoding: otlp_proto
    topic: otel-pod-1
    create_topic: true
    full: true
```
Auto-scaling? ideas...

Rate Sent Spans

Rate Received Spans

Topics

Search by Topic Name

Show Internal Topics

Delete selected topics
Copy selected topic
Purge messages of selected topics

<table>
<thead>
<tr>
<th>Topic Name</th>
<th>Partitions</th>
<th>Out of sync replicas</th>
<th>Replication Factor</th>
<th>Number of messages</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN_consumer_offsets</td>
<td>50</td>
<td>0</td>
<td>1</td>
<td>171</td>
<td>22 KB</td>
</tr>
<tr>
<td>otei-pod-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>83</td>
<td>50 KB</td>
</tr>
<tr>
<td>otei-pod-2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>83</td>
<td>49 KB</td>
</tr>
</tbody>
</table>
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