
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

Balancing Cost and Quality in OpenTelemetry

An Evaluation of Sampling Policies

Zhu Jiekun

About Me

Zhu Jiekun

Quwan

Software Engineer

OpenTelemetry Contributor

Former End User

Current Maintainer

Agenda

01. Quwan’s Distributed Tracing Journey

02. Sampling in OpenTelemetry: Classic Approaches

03. Quantifying Sampling Costs and Quality

04. Tracing Edge Cases: Potential Solutions

05. Build the Future with OpenTelemetry

Drop Your Question

? jiekun.dev/otel

01. Quwan’s Distributed Tracing Journey

分布式追踪在趣丸现状

About Quwan

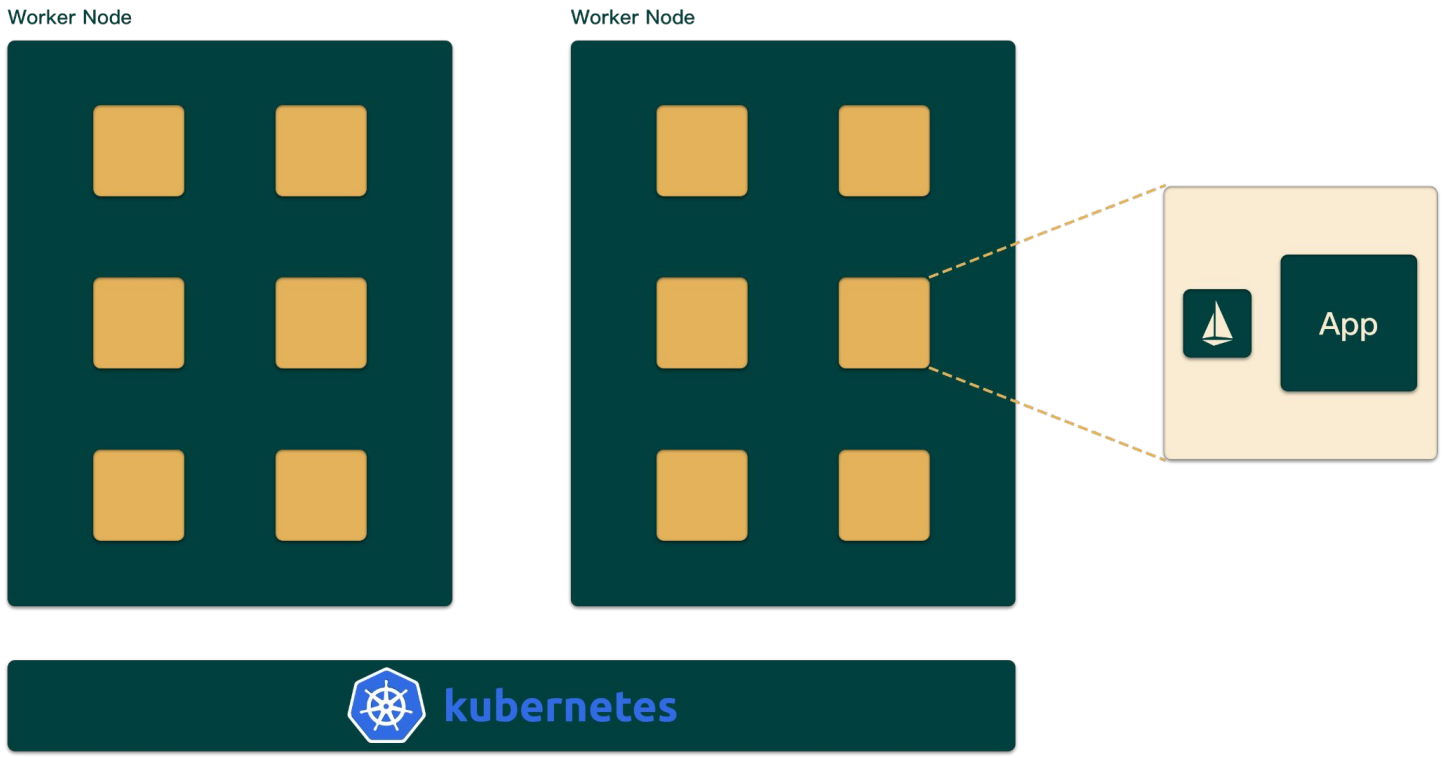
Quwan 趣丸

TT语音

TT电竞

唱鸭

Current Situation

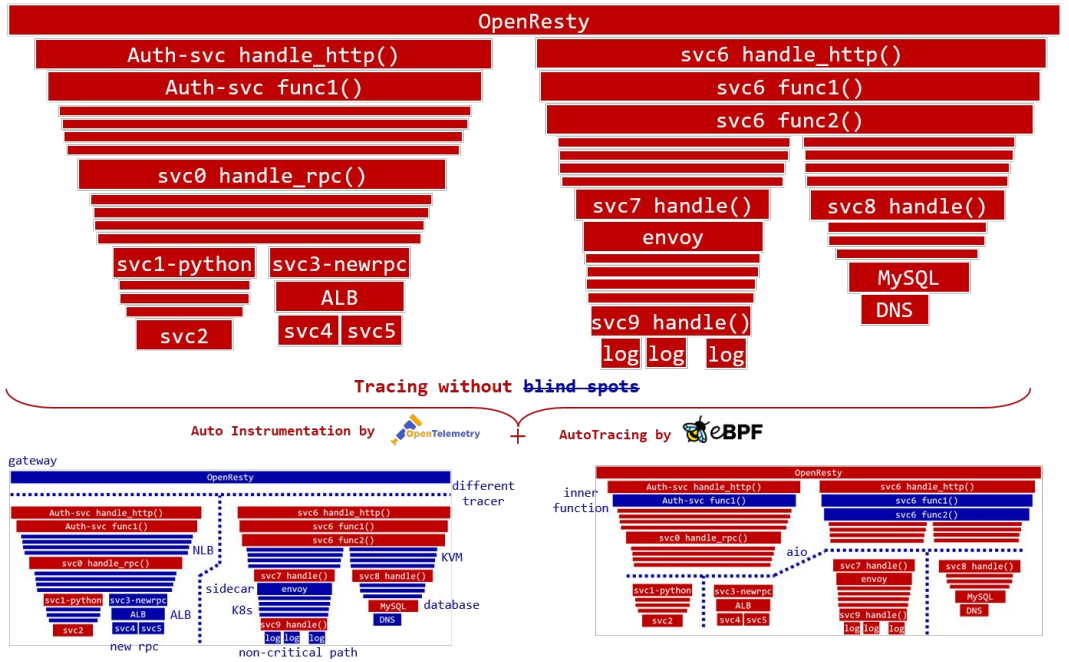


Keypoint

- Build on Kubernetes accompanied by Istio.
  构建在 Kubernetes 之上，搭配 Istio。
- 40+ Billion Spans per Day.
  每日有超过 400 亿 Span 被上报。

What's Next

- Covering Out-of-scope Areas.
  eBPF + DeepFlow 用于覆盖更多场景。



eBPF × DeepFlow®

Questions

- Are all trace data helpful for troubleshooting?
  有多少 Trace 数据对问题排查有帮助？
- How to reduce trace data?
  如何减少 Trace 数据呢？

02. Sampling in OpenTelemetry: Classic Approaches

OpenTelemetry 中的采样方案

How is a Trace Generated

Trace [treɪs]

A Trace represents a call-chain.

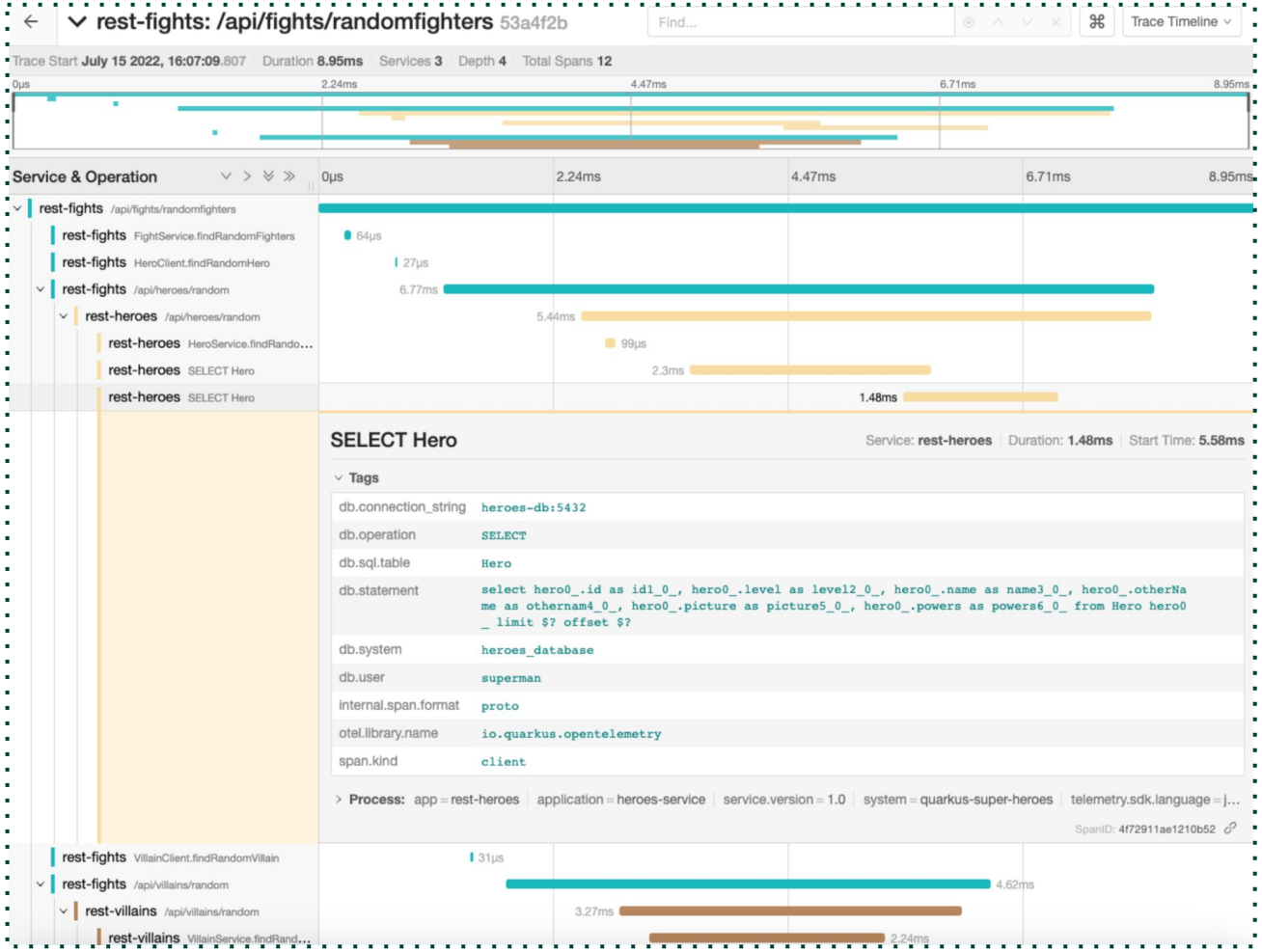
Trace 代表一条调用链路。

Span [spæn]

A Span represents a procedure within a process.

Span 代表一个进程内的的一个过程。

How is a Trace Look Like



Duration (Trace / Span)

span 耗时

Flags (Error / Debug /...)

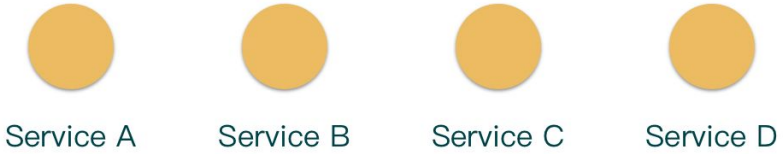
各种标记位

Custom Attributions

自定义属性

Head-based Sampling

Head-based Sampling



Keypoint

- Early decision.

  由**根 Span 决策**，传播至其他 Span。

Pros and Cons

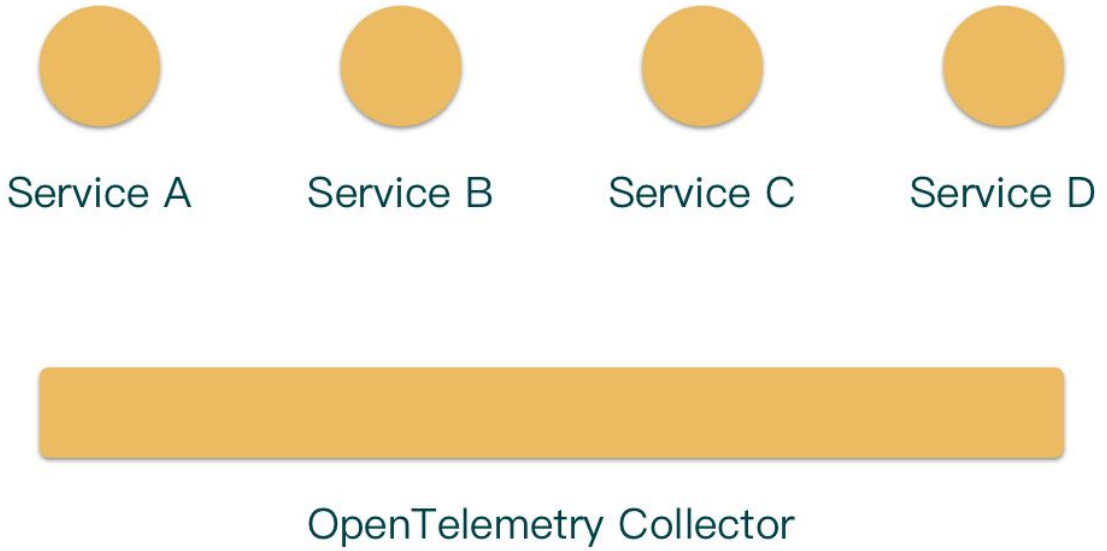
- Lightweight.

  由**应用进行采样**，Trace 平台仅收到少量数据。

- Missing abnormal traces.

  决策时视野有限，错过很多异常情况。

Tail-based Sampling



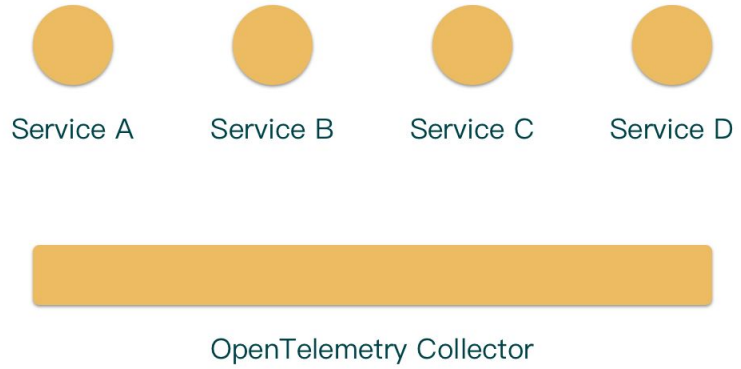
Tail-based Sampling

Keypoint

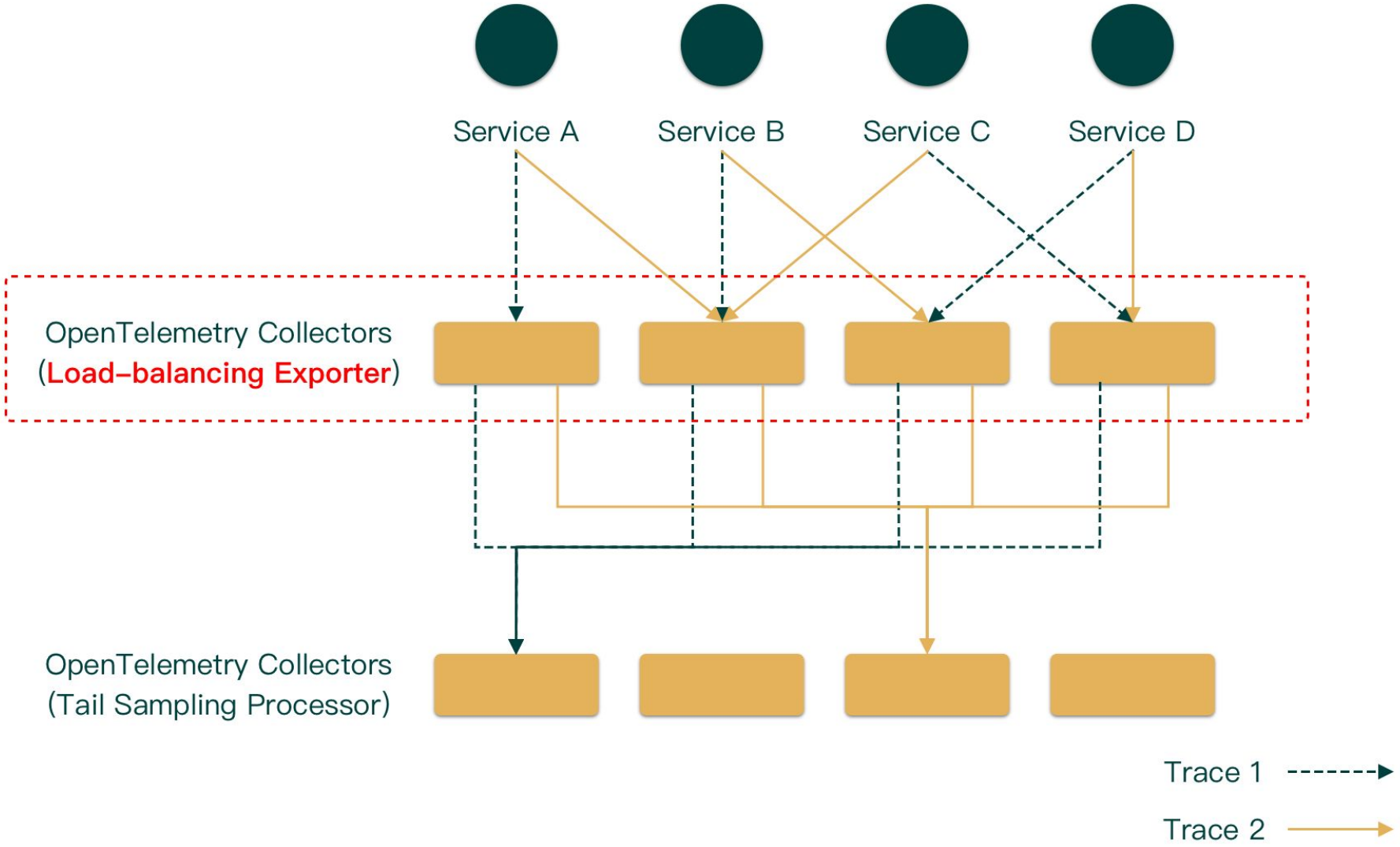
- Postponed decision.
  延迟决策，先收集Trace 信息再判定。

Pros and Cons

- Save disk space while preserving abnormal cases.
  既保留异常情况，又不需要海量的存储资源。
- Require large memory to store tmp data.
  暂存数据多，对基础设施要求高。



Tail-based Sampling: Scaling



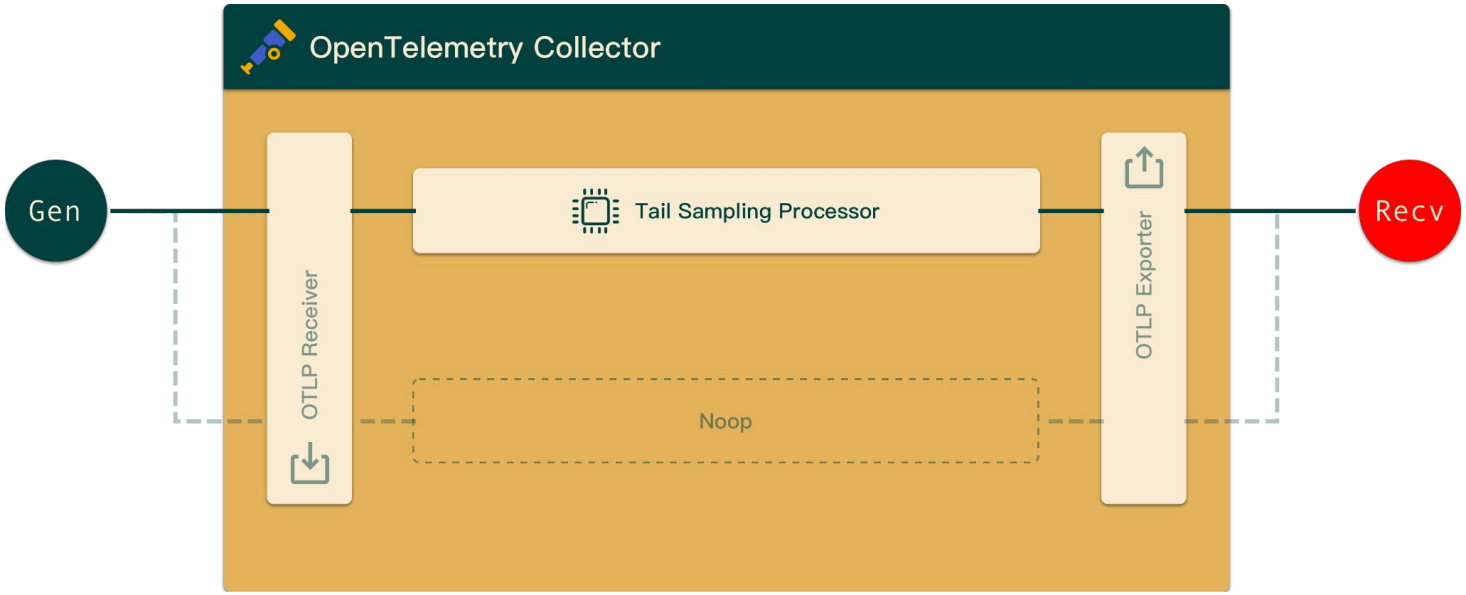
03. Quantifying Sampling Costs and Quality

量化采样成本和效果

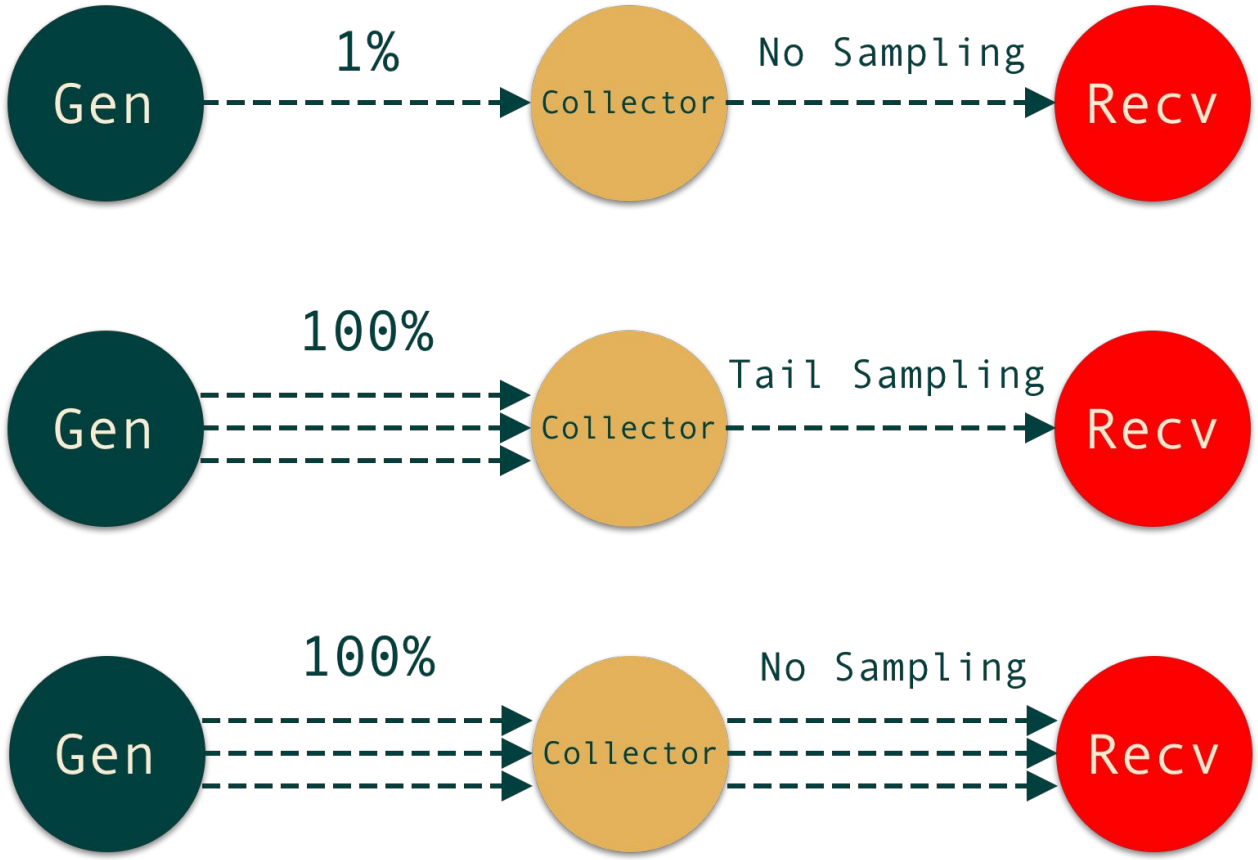
Preparation

Components of the System

- Configurable Trace Generator.
  可配置的 Trace 生成器。
- OpenTelemetry Collector.
  OpenTelemetry Collector 用于数据处理。
- Trace Receiver.
  接收 OTLP 协议 Trace 的 Receiver。
- 4 CPU / 8 GiB Memory.
  Collector 资源使用限制为 4 CPU 和 8 GiB 内存。



Preparation



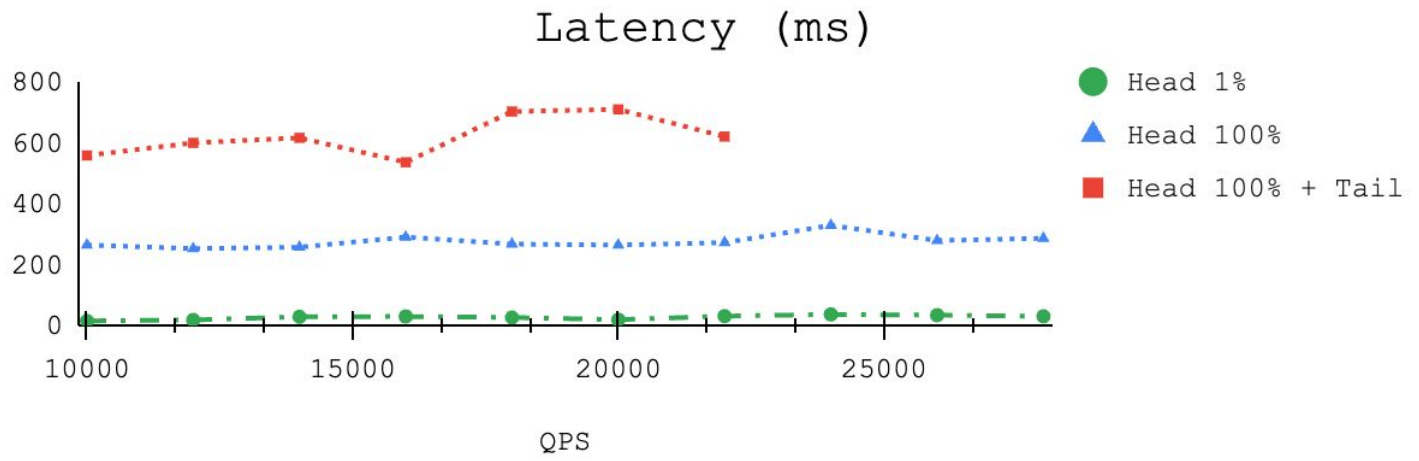
Dataset and Sampling Policies

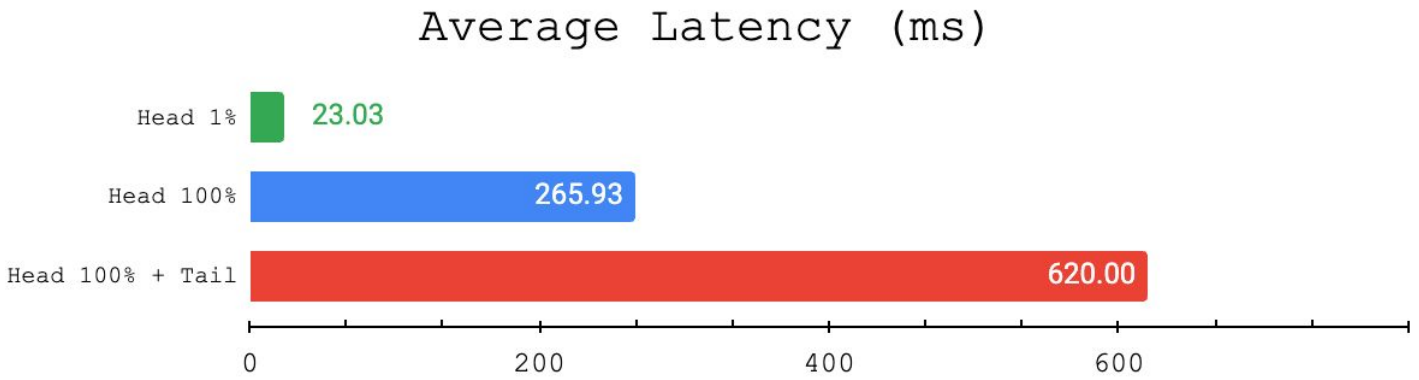
- 400000+ Spans / 20000+ traces.
  平均每组处理 40 万+ Spans / 2 万+ Traces。
- 1% / 100% Head-sampling.
  由 Trace Generator 模拟头部采样，采样率 1% / 100%。
- Tail-sampling by Latency / Error.
  针对长耗时和包含错误的 Trace 进行尾部采样。
- decision_wait: 5s.
  首个 Span 抵达后等待 5 秒评估所属的 Trace。

Benchmark Result

Keypoint

- Significant increase in latency.
  显著处理耗时提升。

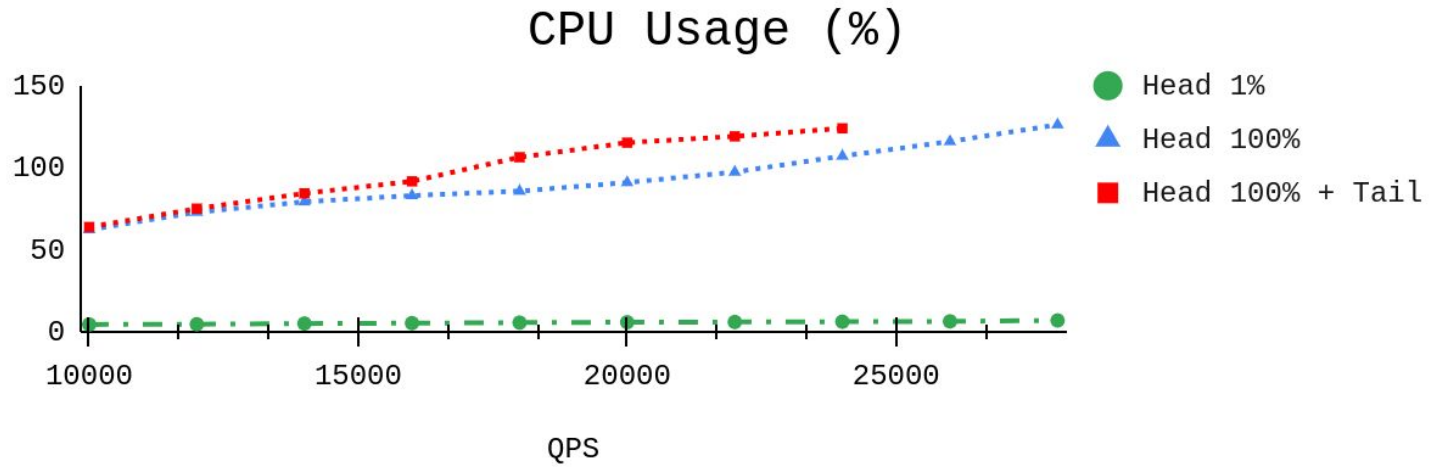


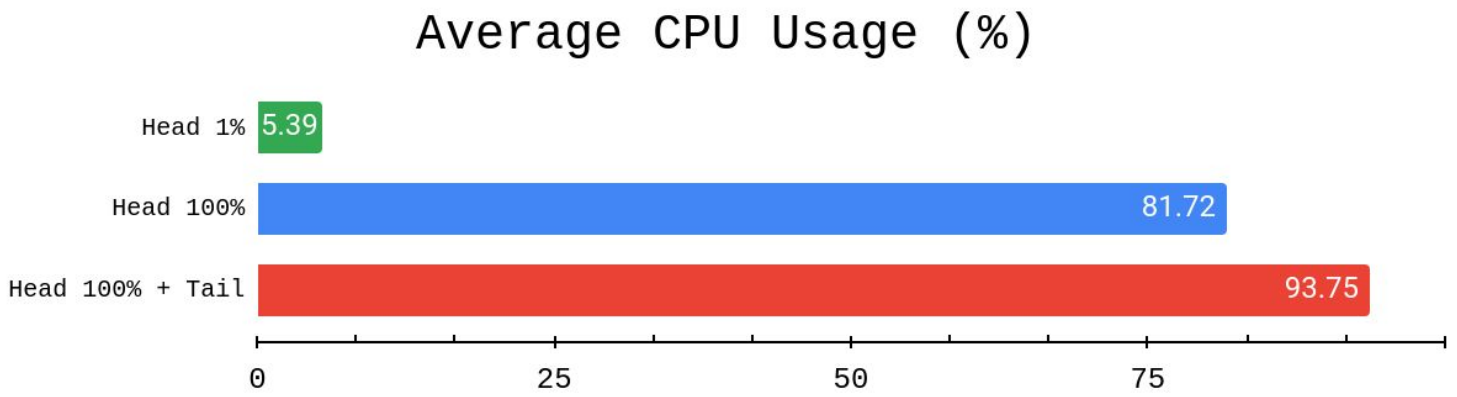


Benchmark Result

Keypoint

- Stronger correlation with the Span number.
  CPU 开销与 Span 上报数量关联更大。
- Weaker correlation with the Collector Processor.
  尾部采样带来 10% 的额外 CPU 开销。

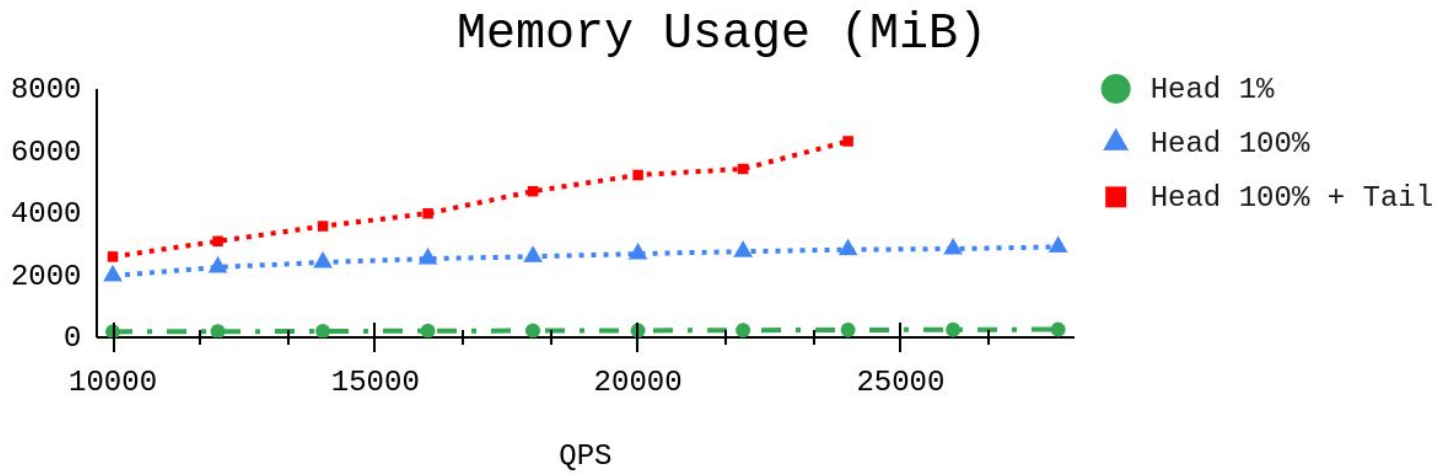


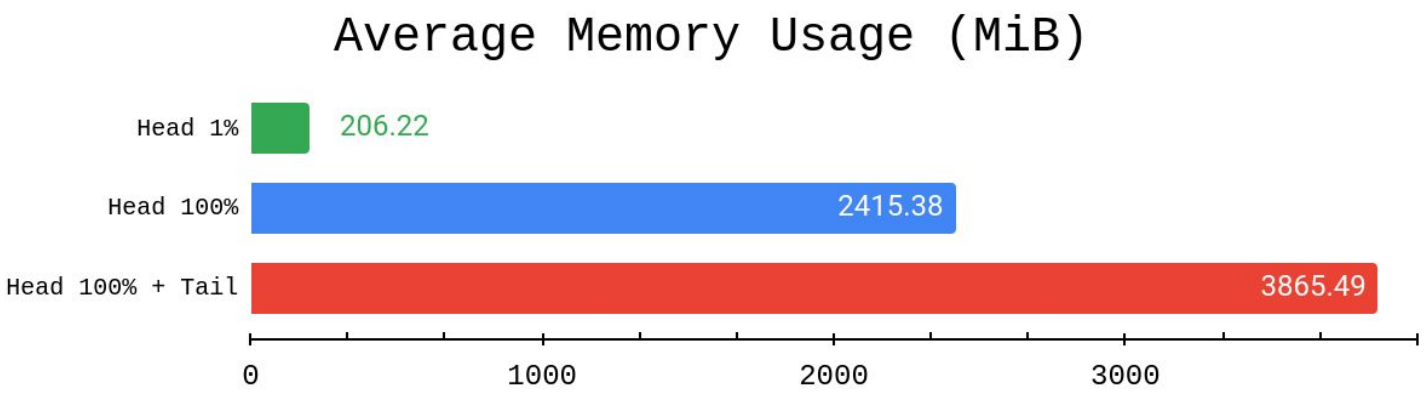


Benchmark Result

Keypoint

- Requiring Infrastructure with large memory.
  尾部采样更青睐于大内存容量的基础设施。
- 20x Mem increment versus Head-Sampling.
  相比头部采样，尾部采样面对 20000 QPS 输入时增加了 20 倍内存使用量。



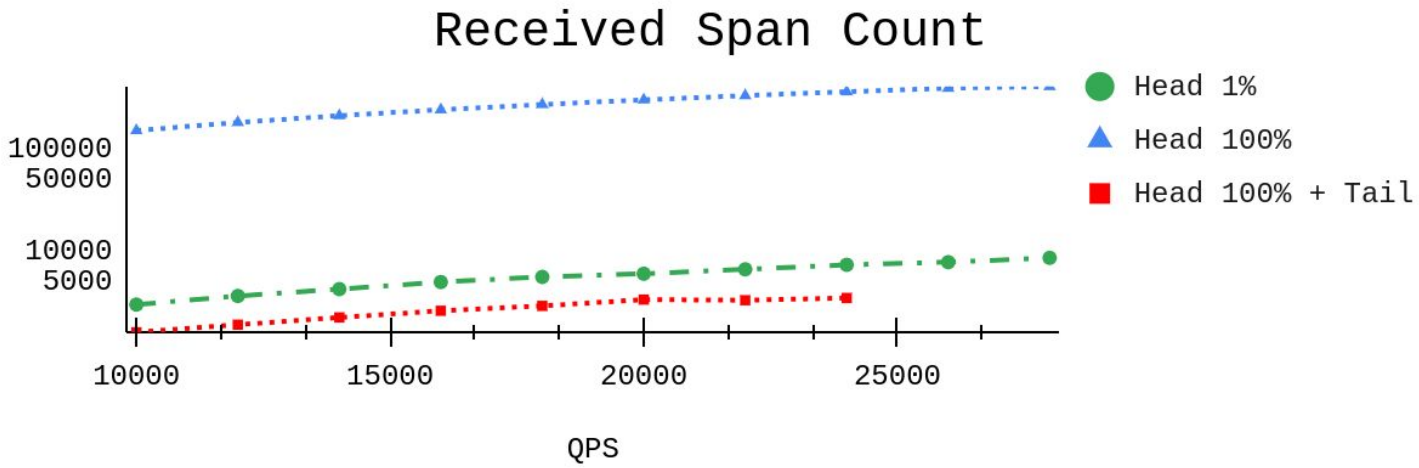


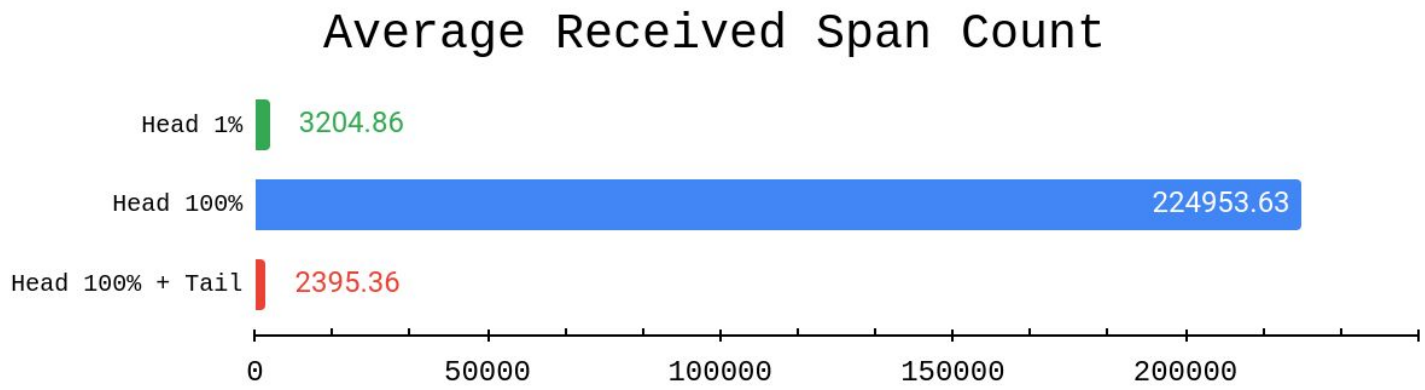
Benchmark Result

Keypoint

- The output data volume of Tail-sampling is highly competitive.

  尾部采样得到的数据量对比 1% 头部采样依然有竞争力及调整空间。





Advices

Recommend

- 1 CPU / 2 GiB for 8000 span per second.
  为每 8000 Span / Sec 的数据压力准备 1 CPU / 2 GiB 内存资源。

Advices

- Provide client SDK with dynamic configuration.
  使用可以由平台方动态调控的 Client SDK。

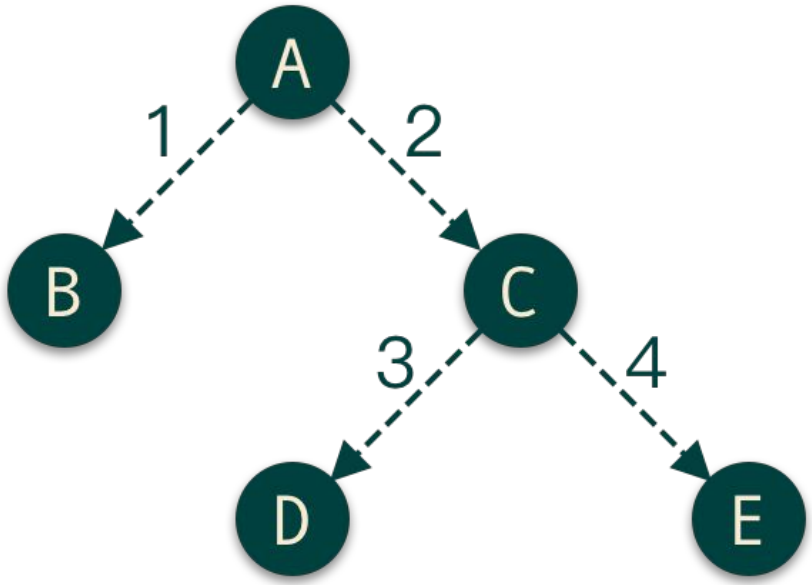
Benefits

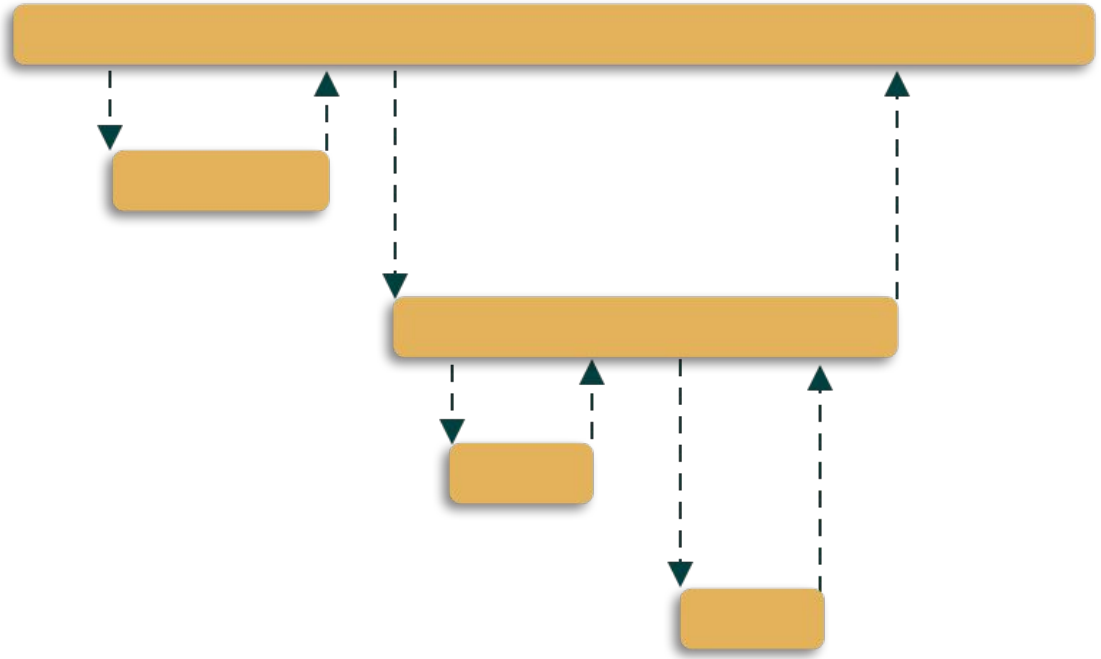
- Real-time awareness.
  对采样率的实时感知，控制风险。
- Dynamic adjustment.
  根据错误率等情况实时动态调整采样率。

04. Tracing Edge Case: Potential Solutions

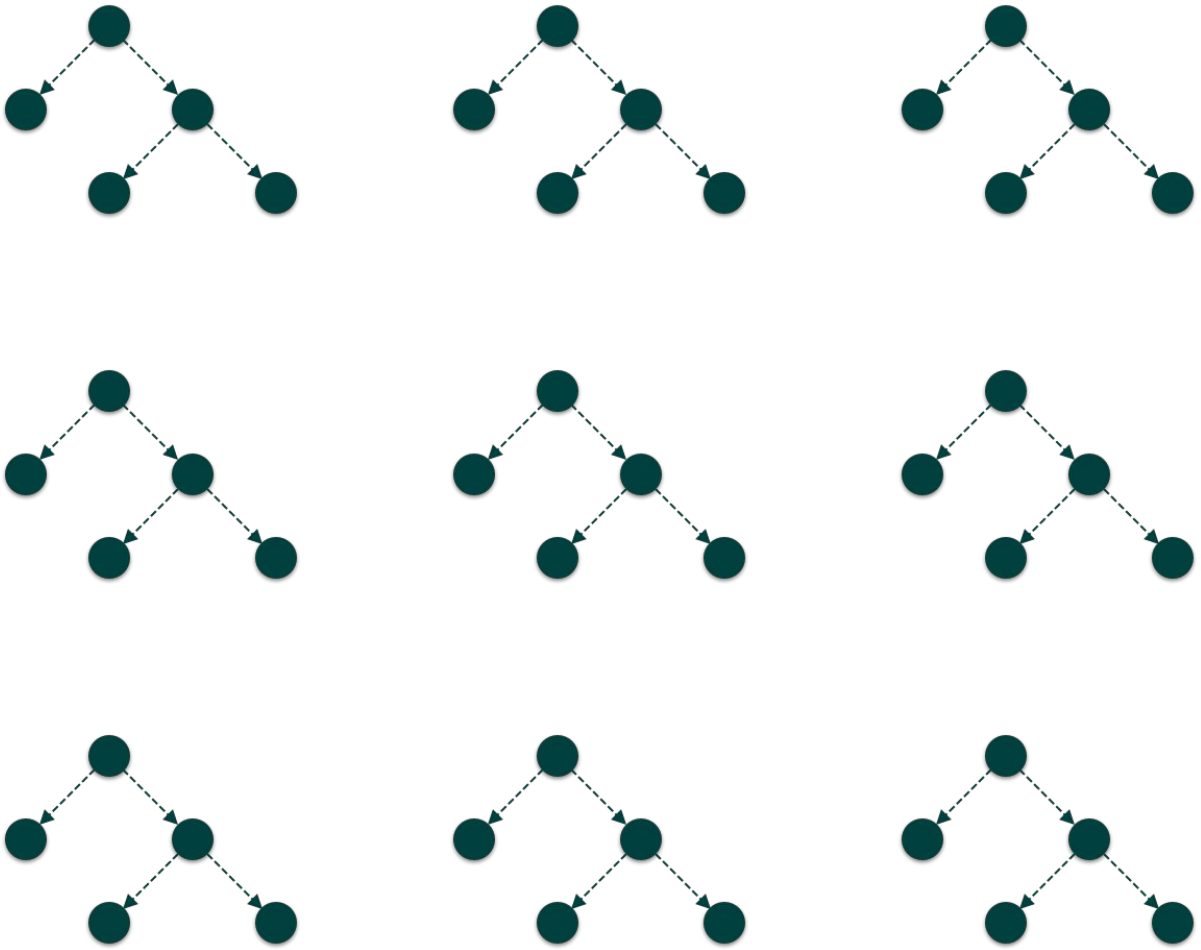
让采样覆盖更多场景

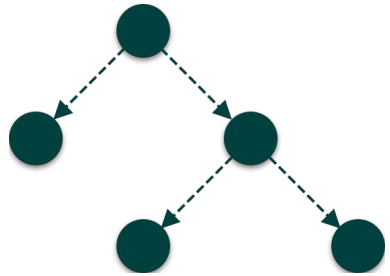
A Trace Example





The Chaos of Trace Data

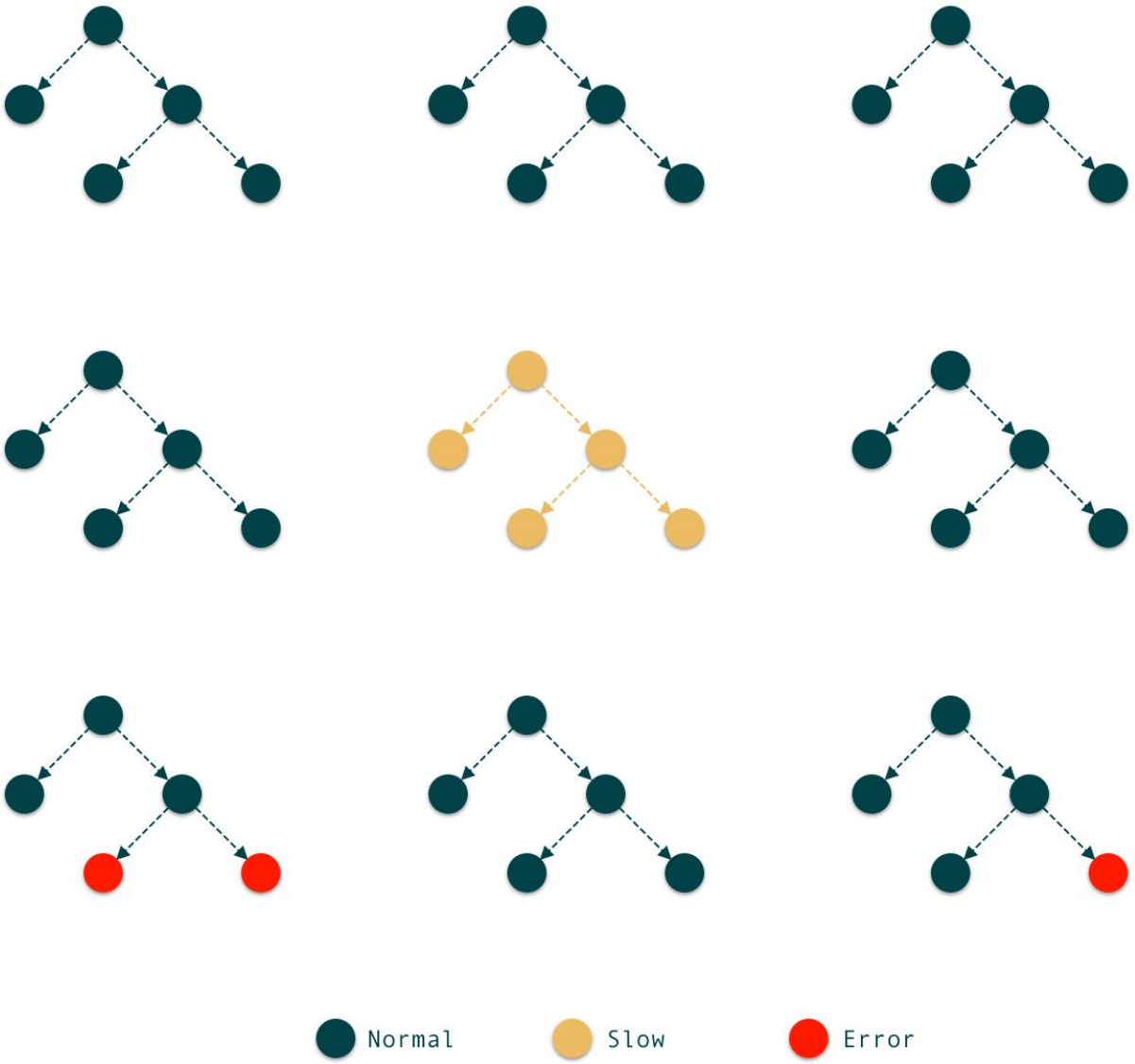


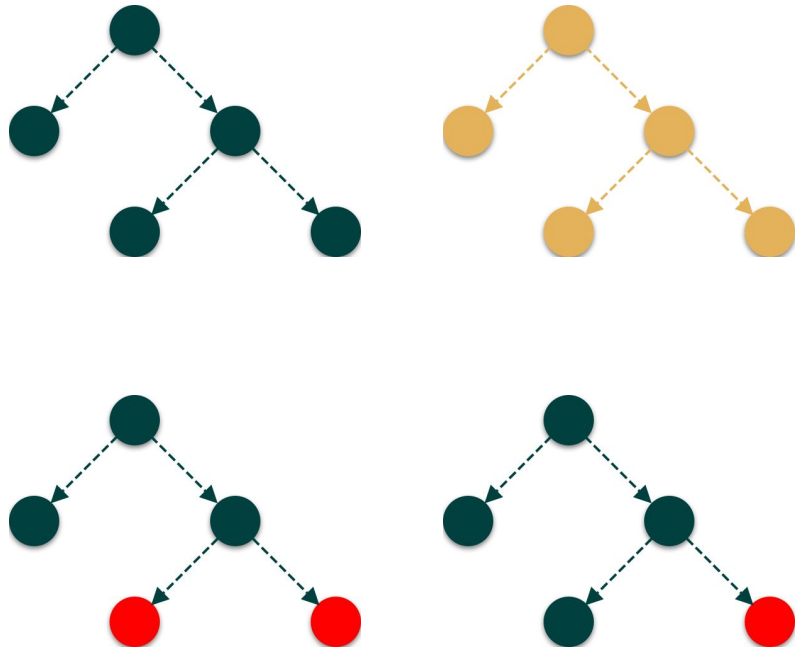


Head-based Sampling Result

头部采样结果

The Chaos of Trace Data



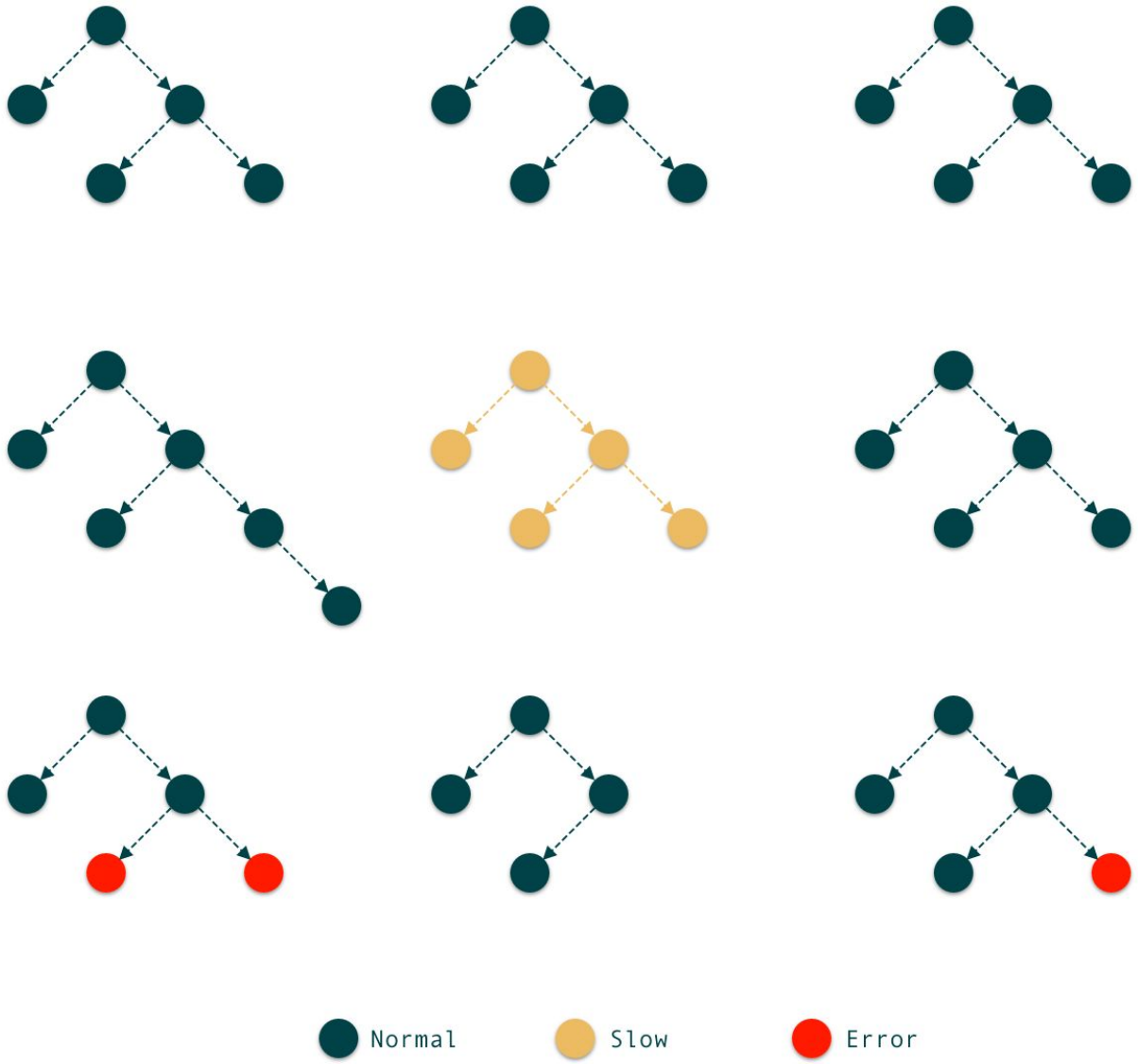


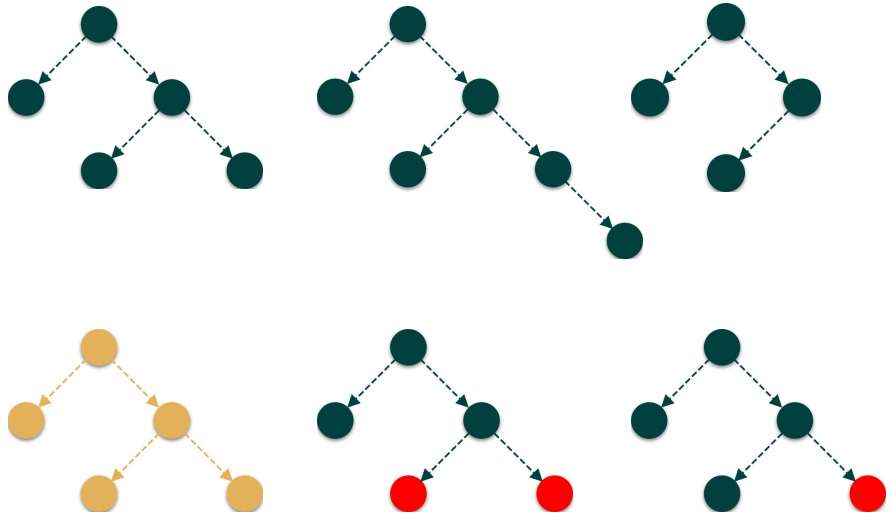
Tail-based Sampling Result

(Based on Error / Latency)

尾部采样结果（基于错误 / 耗时）

The Chaos of Trace Data

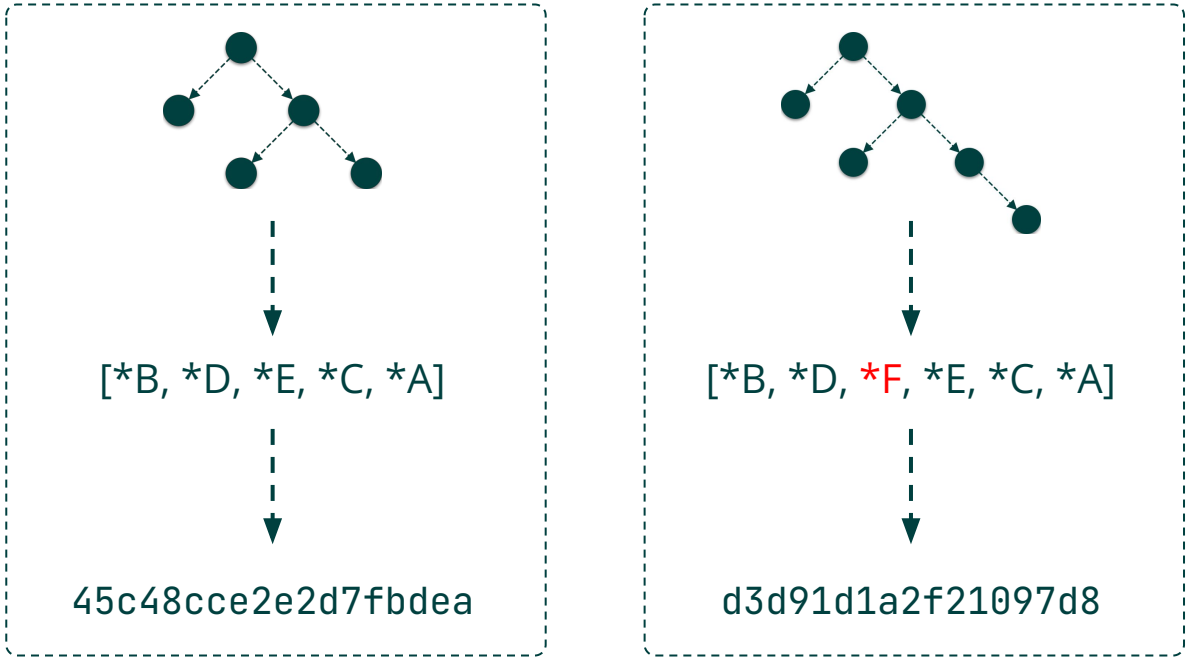




How to Collect **Unique** Traces?

如何捕获一些与众不同的Trace 呢？

Identifying Unique Traces



Metadata Serialization & Digestion

元数据序列化 & 摘要计算

Trace ID	Type ID	Frequency (/min)	Sample Decision
c4ca42	45c4	9238212	No
c81e72			No
eccbc8			No
a87ff6	d3d9	8	Yes

Sample Decision

采样决策

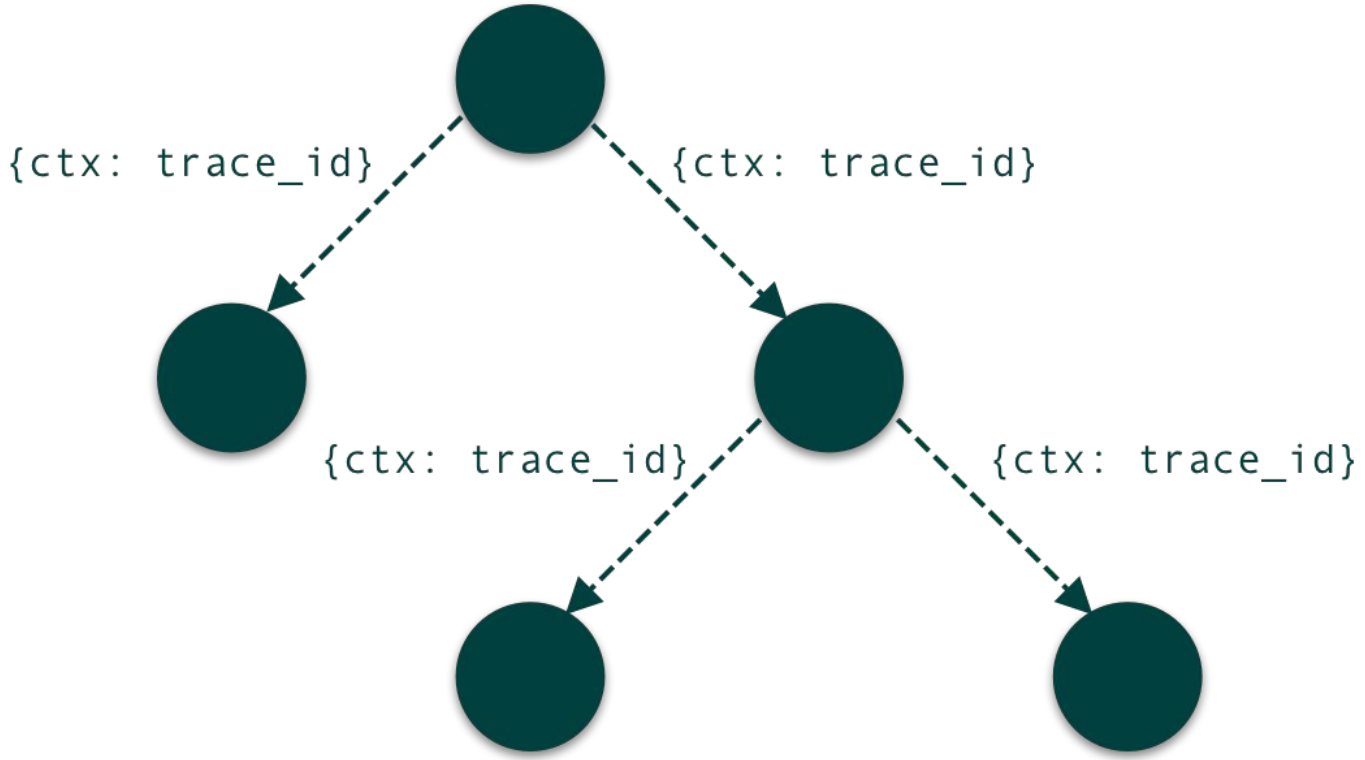
Minimizing Data Transfer

Current Status:

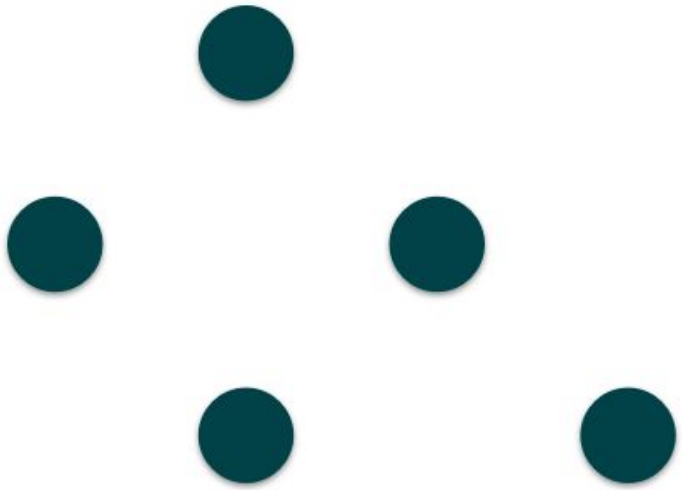
- Trace context propagate from top to bottom.
  当前，Trace 信息从上往下传播。

What If:

- Trace context propagate from bottom to top.
  Trace 信息从下往上传播会怎么样？



Retroactive Sampling



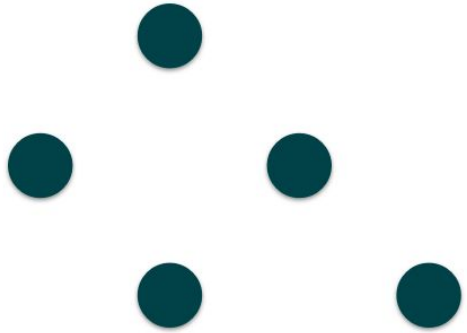
Retroactive Sampling

Pros

- Grant all applications control over sampling.
  让所有应用服务都拥有采样的决定权
- Less data being reported by client SDK.
  Client 侧采样，更少数据上报至后续处理流程。

Cons

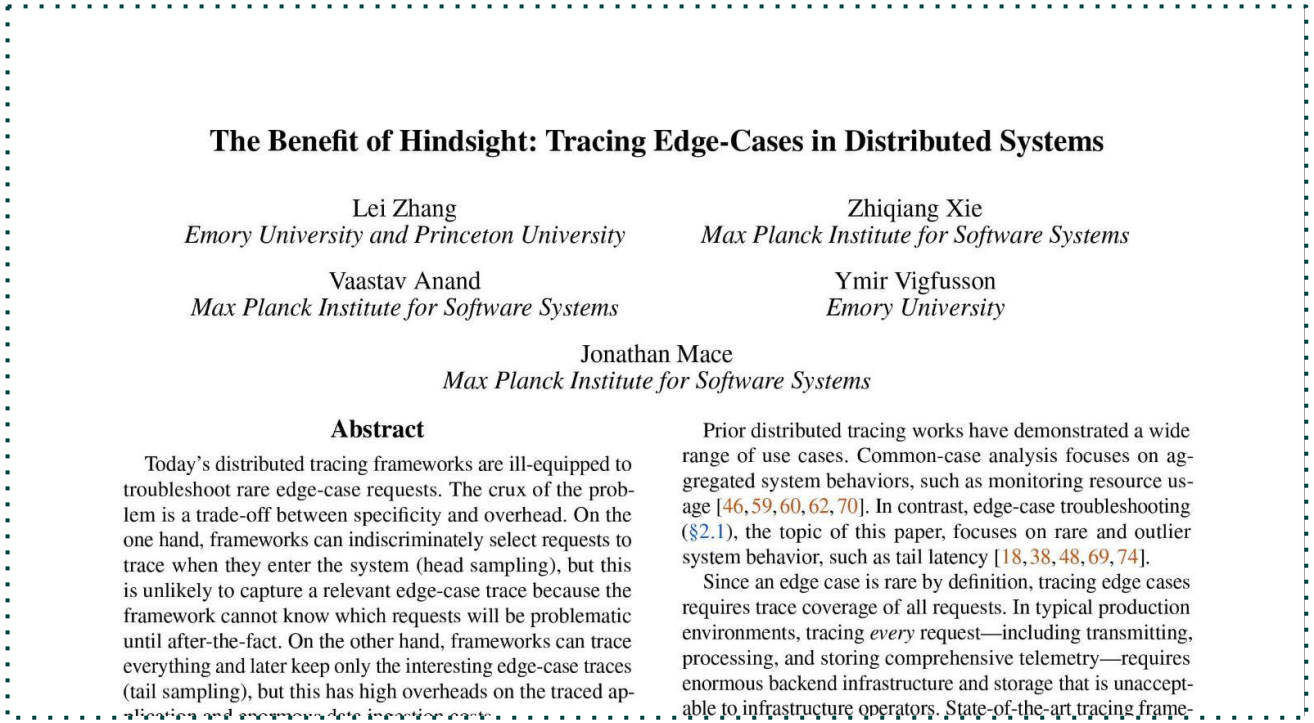
- Completed spans cannot be sampled.
  已结束的 Span 无法被采样到。
- Require a highly unified RPC framework / protocol.
  依赖高度统一的通信框架或协议。



Retroactive Sampling: Further Reading

Goals

- Applications control.
  应用服务决策。
- Low traffic.
  低数据传输量。
- No dependency on frameworks or SDKs.
  不依赖于特定框架或SDK。
- Coherent sampling.
  采样数据保持连贯，不缺失 Span。



The Benefit of Hindsight: Tracing Edge-Cases in Distributed Systems

Lei Zhang
Emory University and Princeton University

Zhiqiang Xie
Max Planck Institute for Software Systems

Vaastav Anand
Max Planck Institute for Software Systems

Ymir Vigfusson
Emory University

Jonathan Mace
Max Planck Institute for Software Systems

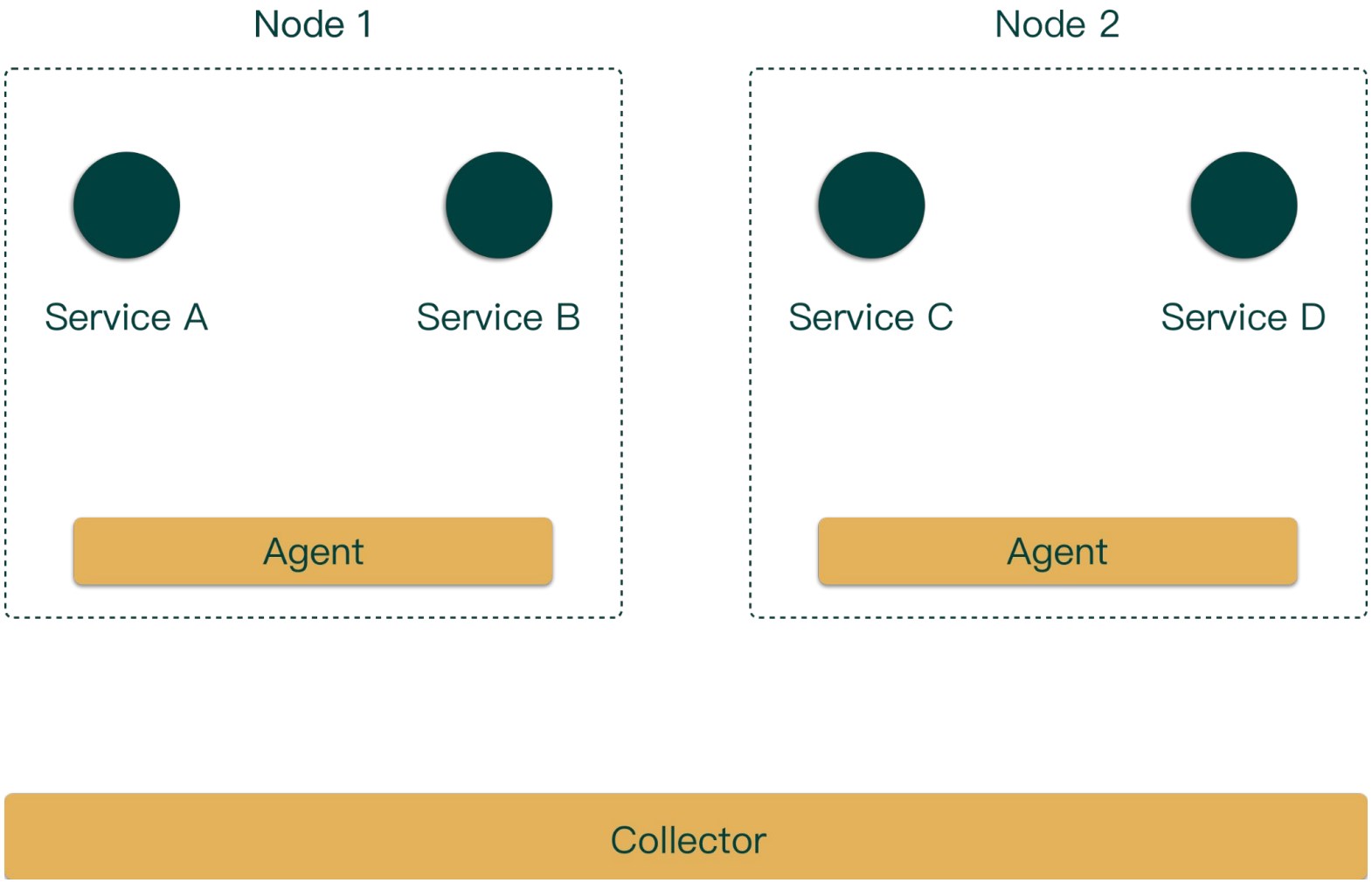
Abstract

Today's distributed tracing frameworks are ill-equipped to troubleshoot rare edge-case requests. The crux of the problem is a trade-off between specificity and overhead. On the one hand, frameworks can indiscriminately select requests to trace when they enter the system (head sampling), but this is unlikely to capture a relevant edge-case trace because the framework cannot know which requests will be problematic until after-the-fact. On the other hand, frameworks can trace everything and later keep only the interesting edge-case traces (tail sampling), but this has high overheads on the traced applications and enormous data ingestion costs.

Prior distributed tracing works have demonstrated a wide range of use cases. Common-case analysis focuses on aggregated system behaviors, such as monitoring resource usage [46, 59, 60, 62, 70]. In contrast, edge-case troubleshooting (§2.1), the topic of this paper, focuses on rare and outlier system behavior, such as tail latency [18, 38, 48, 69, 74].

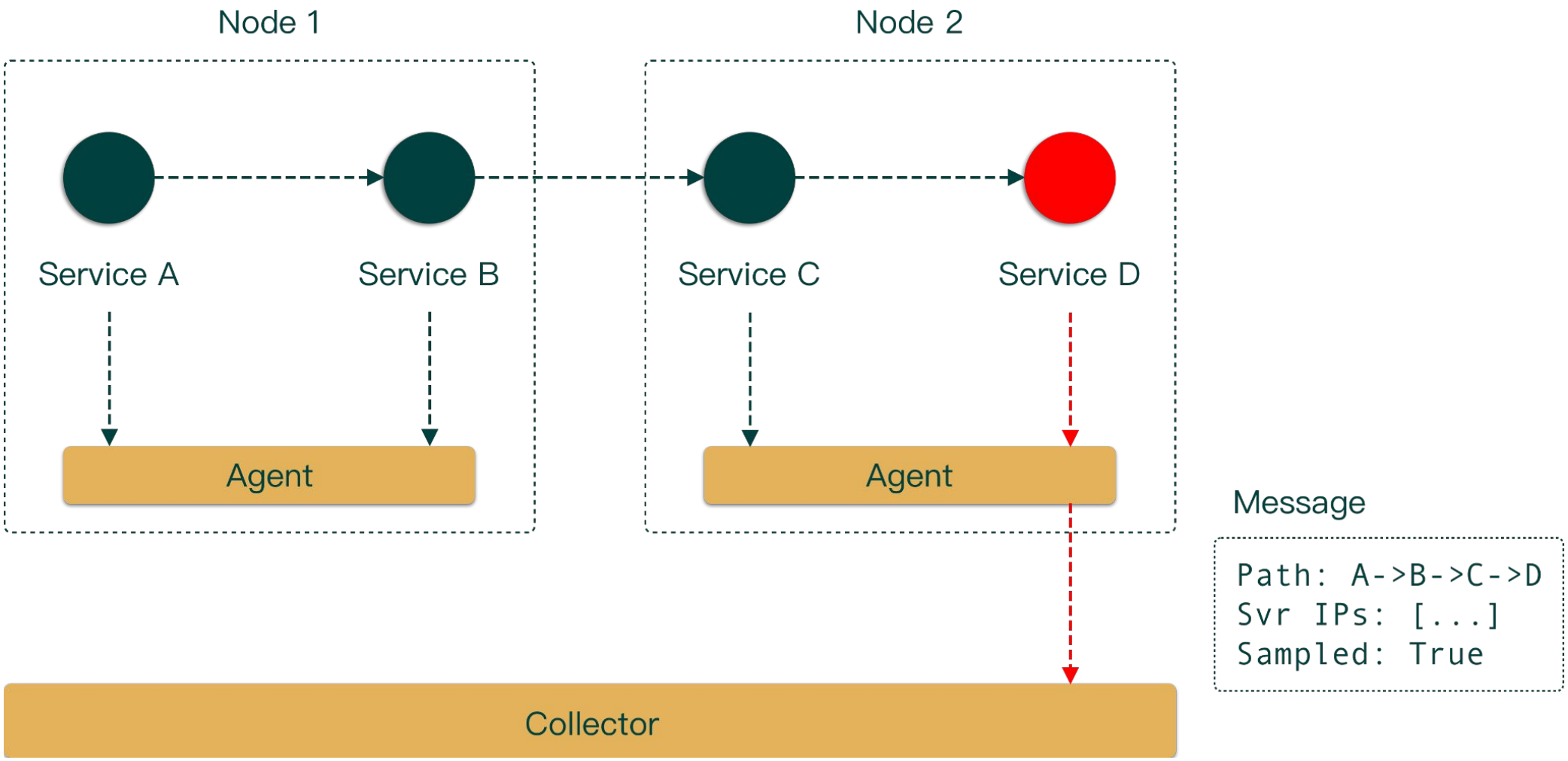
Since an edge case is rare by definition, tracing edge cases requires trace coverage of all requests. In typical production environments, tracing *every* request—including transmitting, processing, and storing comprehensive telemetry—requires enormous backend infrastructure and storage that is unacceptable to infrastructure operators. State-of-the-art tracing frame-

Retroactive Sampling: Further Reading

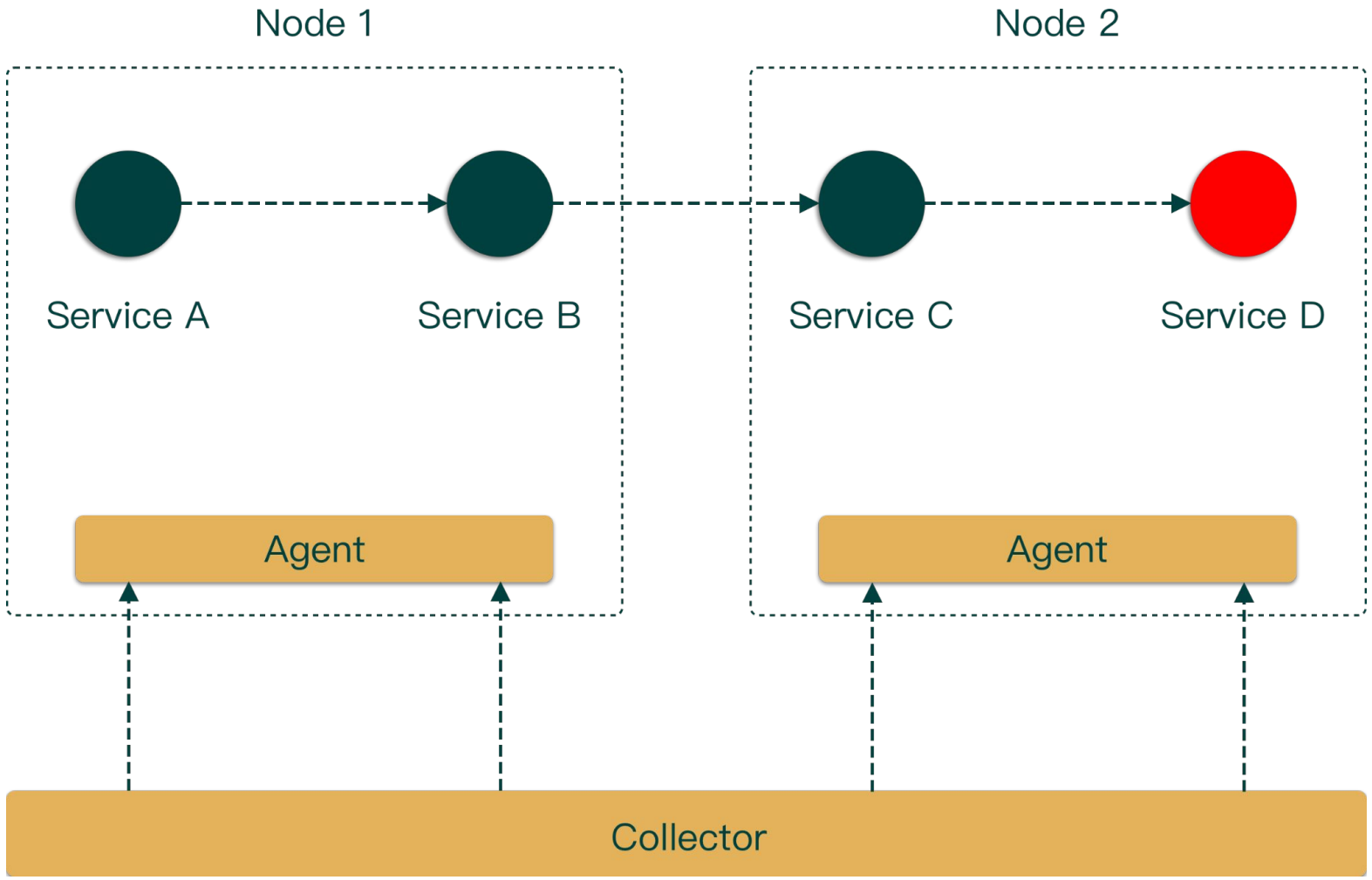


Retroactive Sampling: Further Reading

jiekun.dev/otel



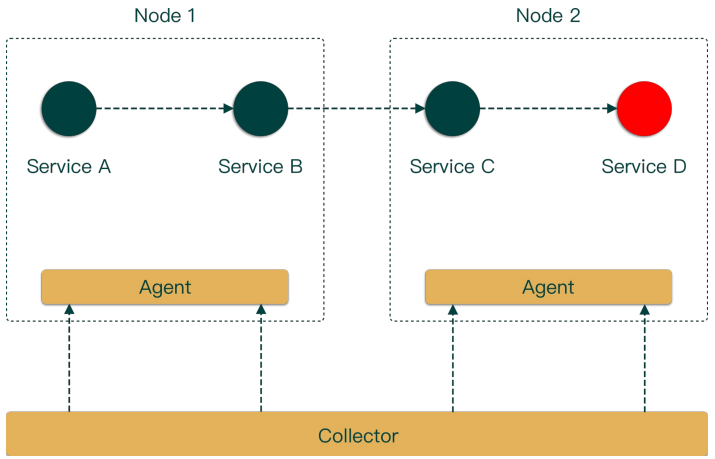
Retroactive Sampling: Further Reading



Retroactive Sampling: Further Reading

Keypoint

- Only metadata is transmitted over the network for the majority of the time.
  大多数时间只有 metadata 在网络中传输，降低数据量。
- Address info is included in the metadata as breadcrumbs and sent to the collector.
  调用链路信息以面包屑形式随metadata 发送给 Collector。
- The collector requests full trace data only when an issue occurs.
  仅当有问题发生时，Collector 才向 Agent 索要完整数据。

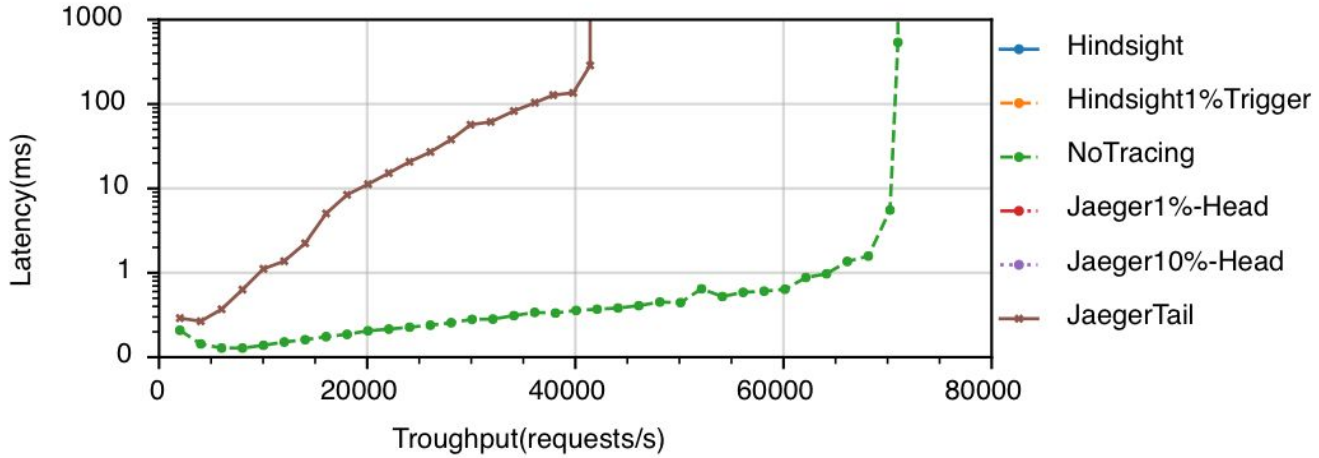


Retroactive Sampling: Further Reading

TL;DR:

- Impact <3.5% when 200 MB/s trace data generated.

  在每秒产生 200 MB Trace 数据的情况下，对延迟和吞吐量影响不足 3.5%。

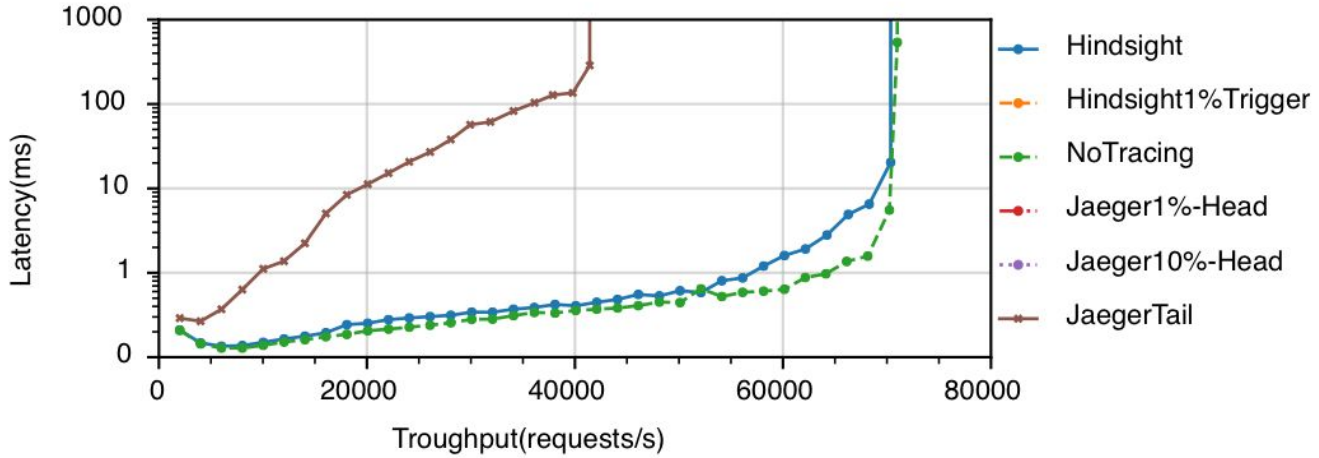


Retroactive Sampling: Further Reading

TL;DR:

- Impact <3.5% when 200 MB/s trace data generated.

  在每秒产生 200 MB Trace 数据的情况下，对延迟和吞吐量影响不足 3.5%。

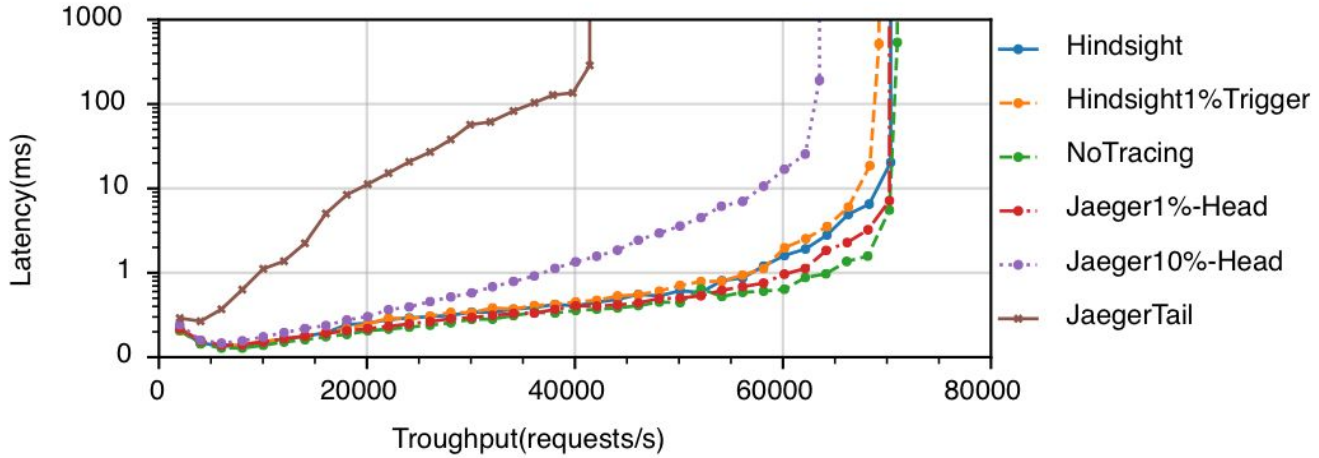


Retroactive Sampling: Further Reading

TL;DR:

- Impact <3.5% when 200 MB/s trace data generated.

  在每秒产生 200 MB Trace 数据的情况下，对延迟和吞吐量影响不足 3.5%。



Retroactive Sampling: Further Reading

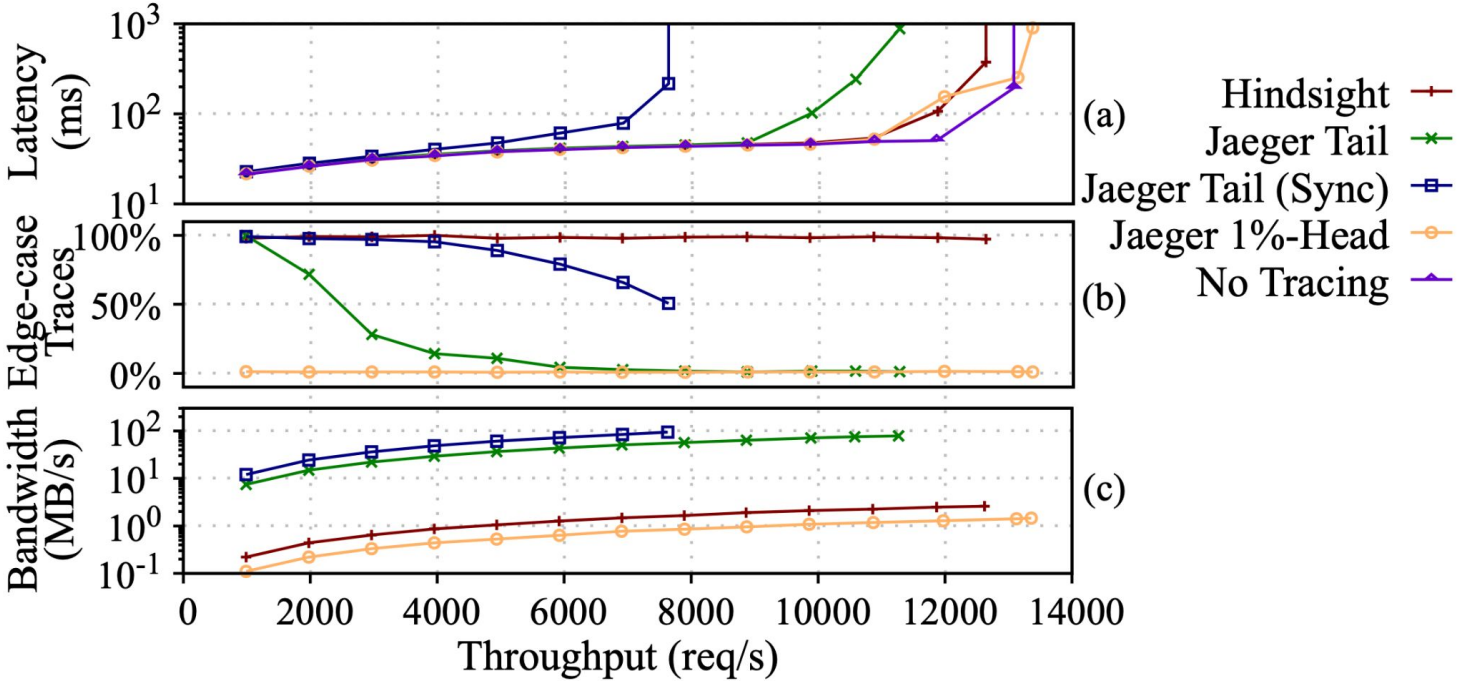
jiekun.dev/otel

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2023

TL;DR:

- Impact <3.5% when 200 MB/s trace data generated.

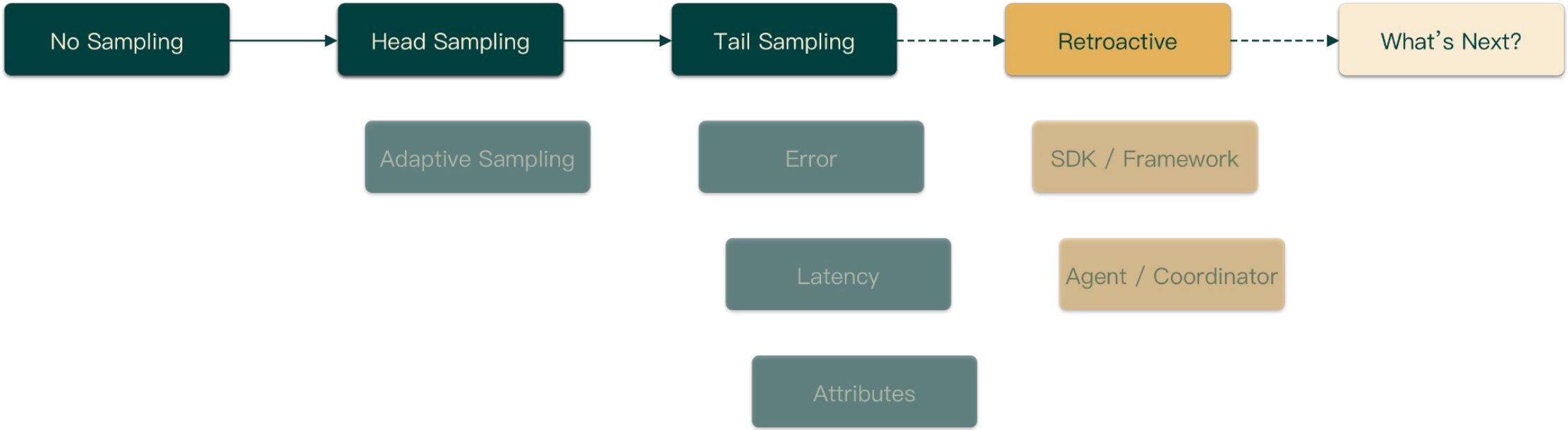
  在每秒产生 200 MB Trace 数据的情况下，对延迟和吞吐量影响不足 3.5%。



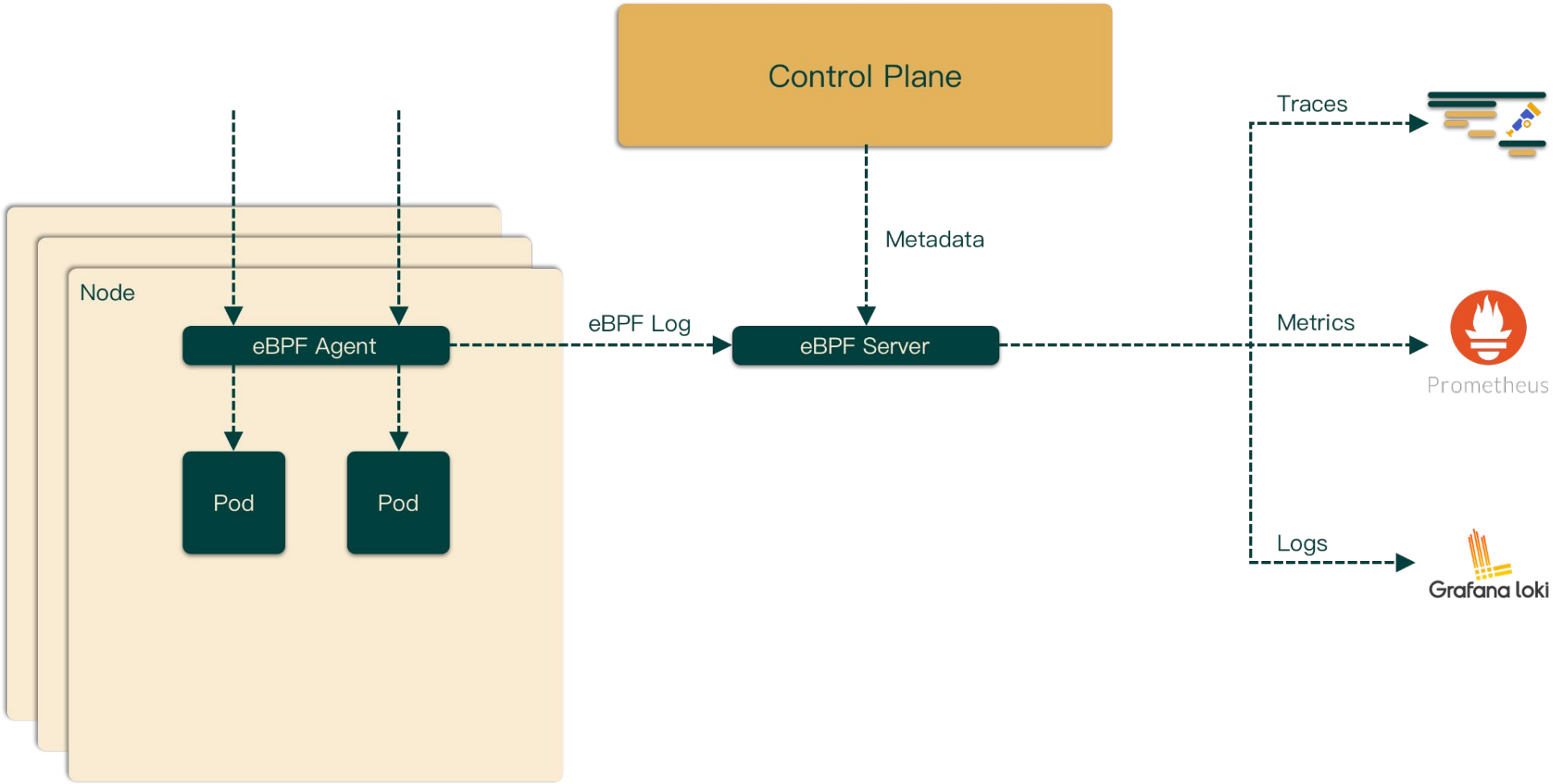
05. Build the Future with OpenTelemetry

杂谈 & 分享你的想法

Sampling Strategy Evolution

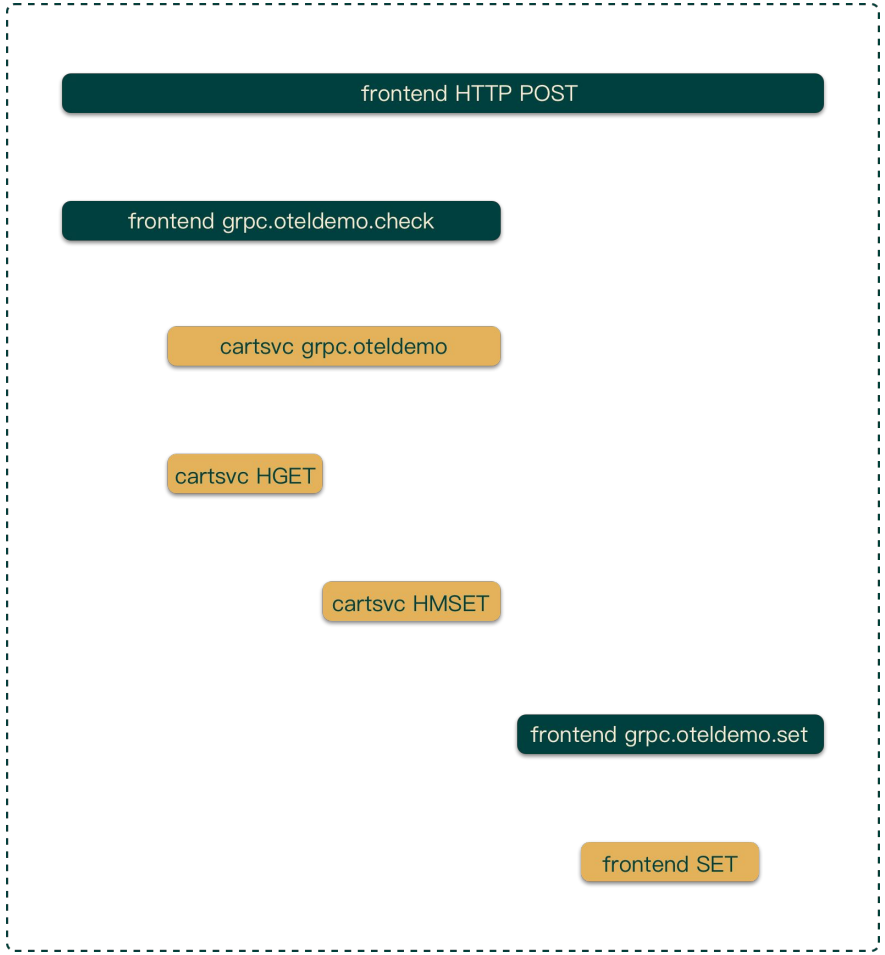


About Auto-Instrumentation

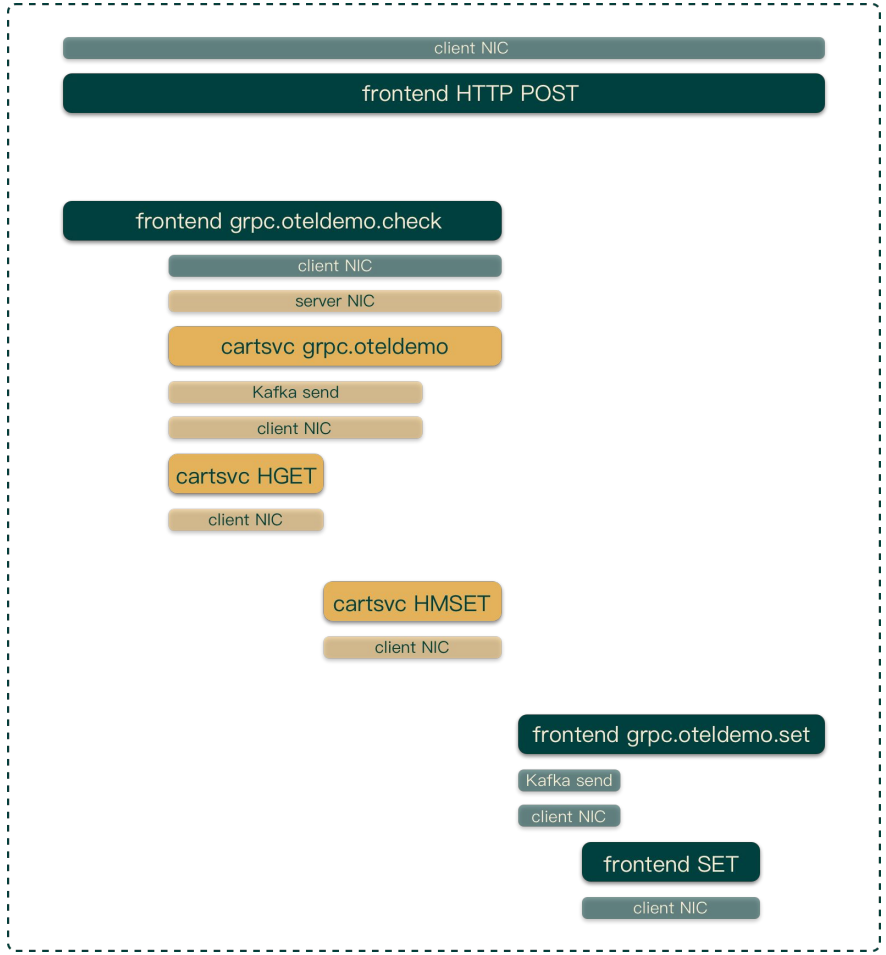


About Auto-Instrumentation

Manual Instrumentation



eBPF



About Auto-Instrumentation

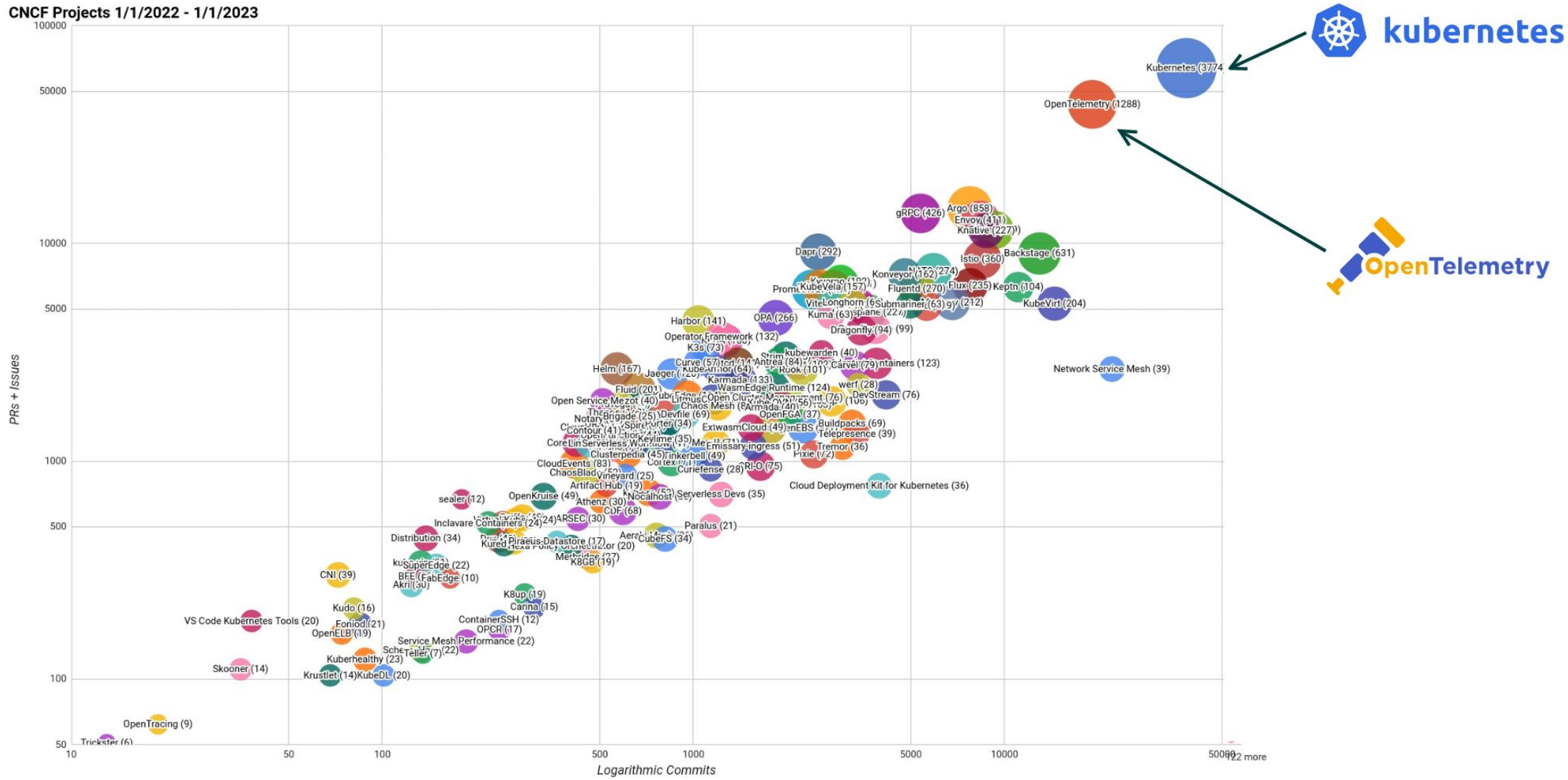

```
func Exec(ctx Context, sql string) (result interface{}) {


  return db.WithContext(ctx).Exec(sql)
}
```

OpenTelemetry Community

jiekun.dev/otel

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2023




KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023

Thank You for Listening


TailSampling


Jiekun

jiekun.dev/otel