Grafana Labs

そのテレメトリーサンプリング コスト改善になっていますか

Cloud Operator Days Tokyo 2026 – July 2026

山口 能迪 (@ymotongpoo)
Staff Developer Advocate, Grafana Labs

Grafana Labs

山口 能迪（やまぐちよしふみ）

Staff Developer Advocate
Grafana Labs

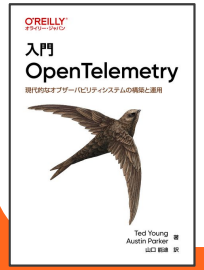
専門領域

- オブザーバビリティ
- SRE

@ymotongpoo

トレースはとても大事

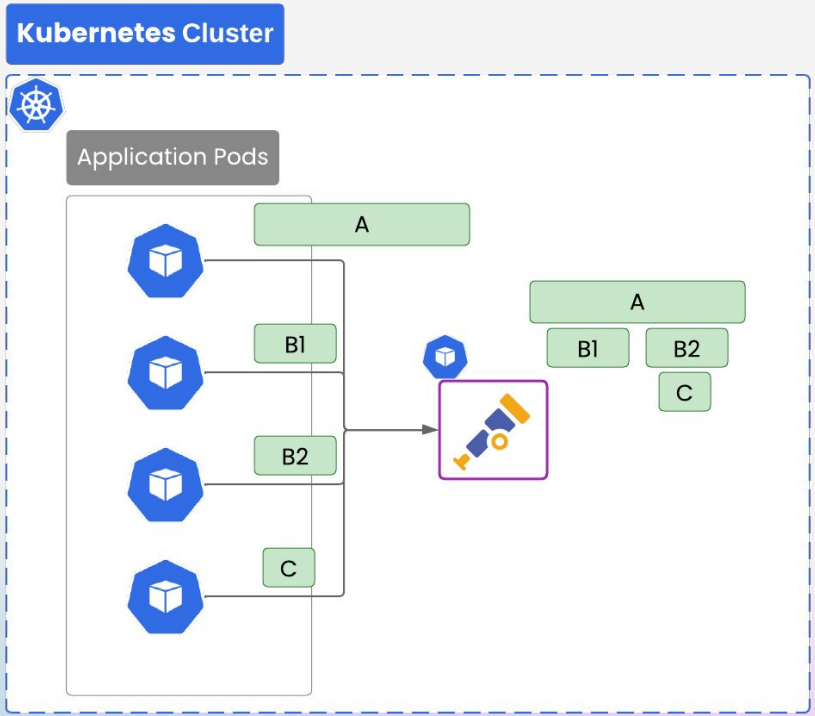
トレースはトランザクションのオブザーバビリティの中核です。
分散システムのパフォーマンス、健全性、そして本番環境での挙動を理解するための最善の方法です。



入門OpenTelemetry | 3章 OpenTelemetry 概要

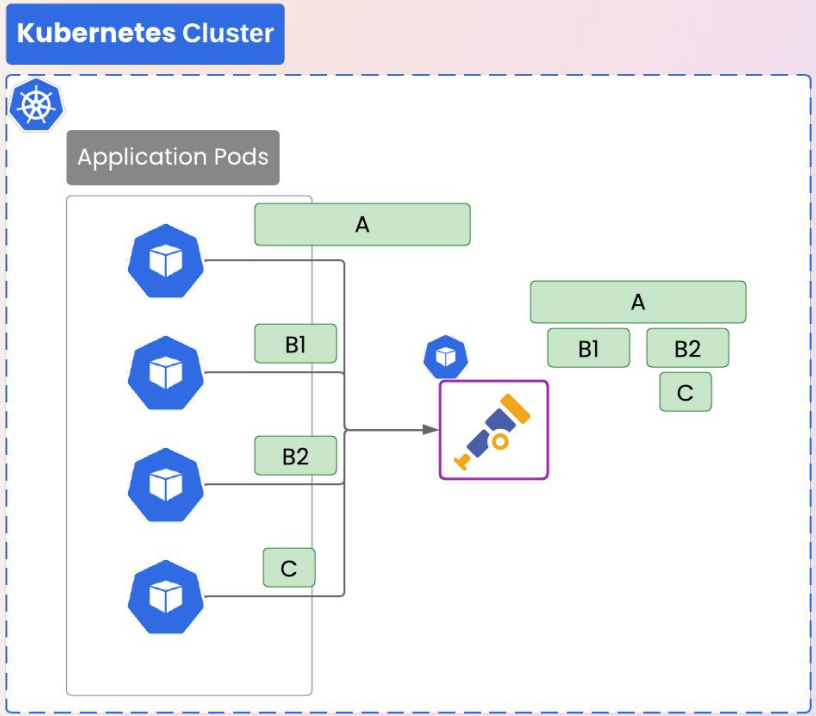
シナリオ: 分散トレースを取得したい

分散トレースを取得する計装をした



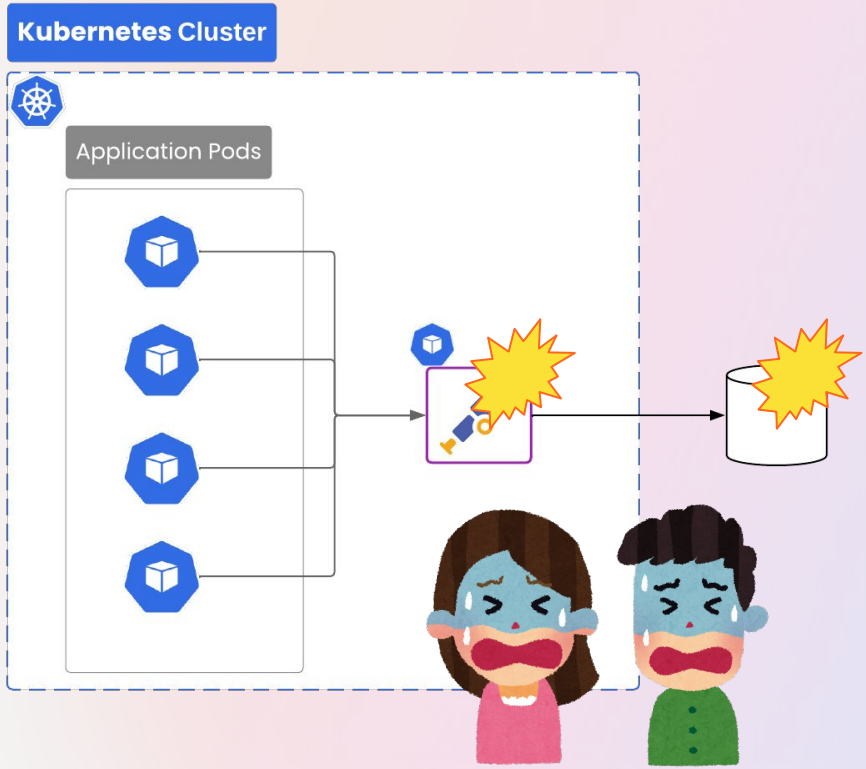
最初期：ボトルネック発見

- スロークエリが見つかった
- 不必要な直列処理を発見した
- 予期しない依存関係を見つけた



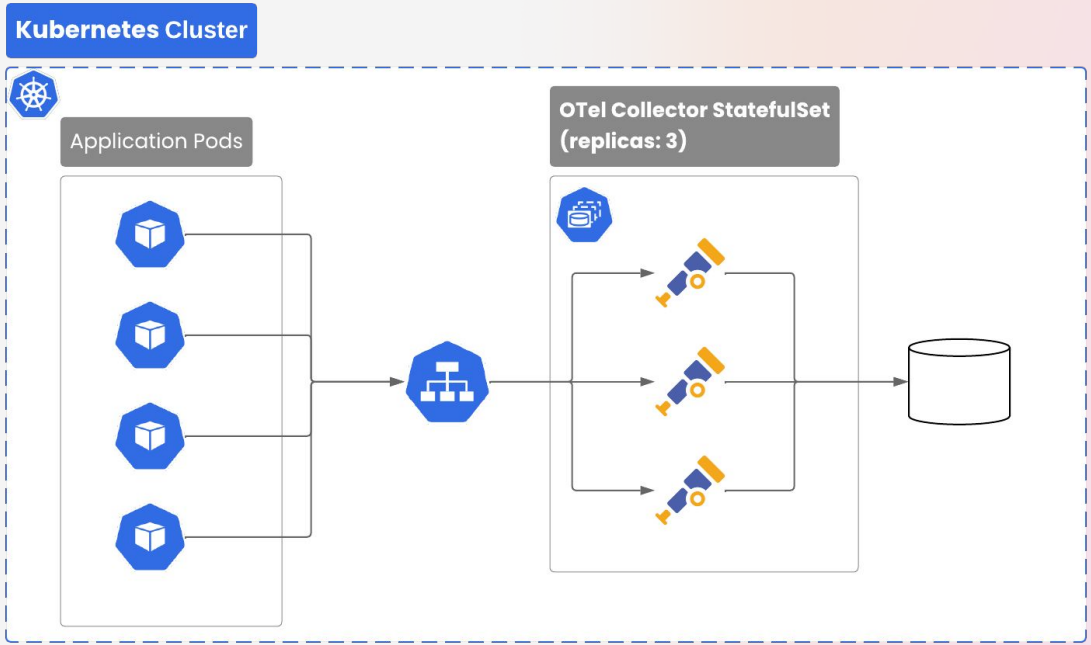
中期: トレース過多

- ストレージコスト高騰
- 分析ツールの負荷増
- 計測時のオーバーヘッド増
- コレクターのSPOF化

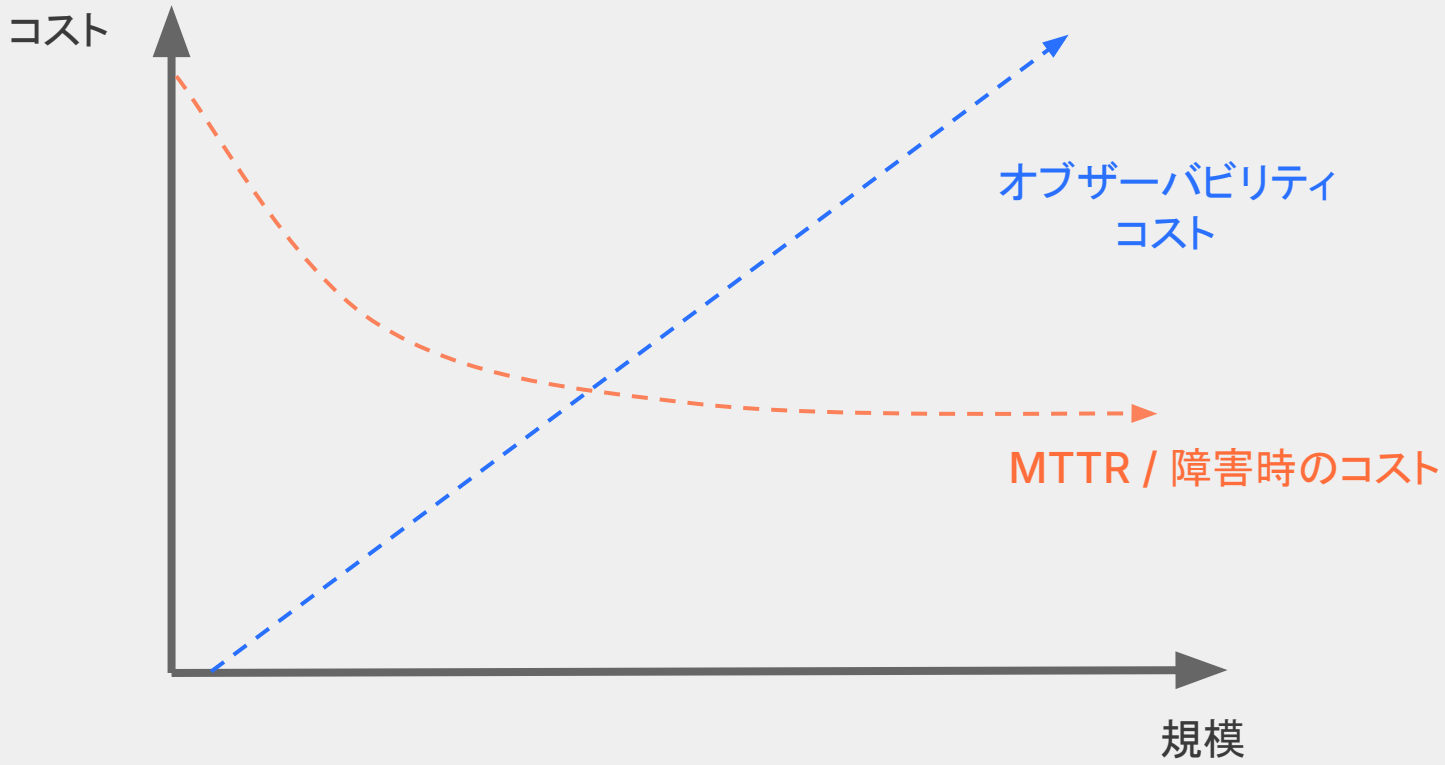


コレクターを複数立てて負荷分散

解決策 1: コレクタープール



多くのデータ ≠ 良い結果



トレースのサンプリングの大きな分岐点

解決策 2: サンプリング

ヘッドベースサンプリング

最初のスパンを受け取った瞬間に記録有無を決定

確率的サンプリング
- eg) 100回に1回記録 (1%)

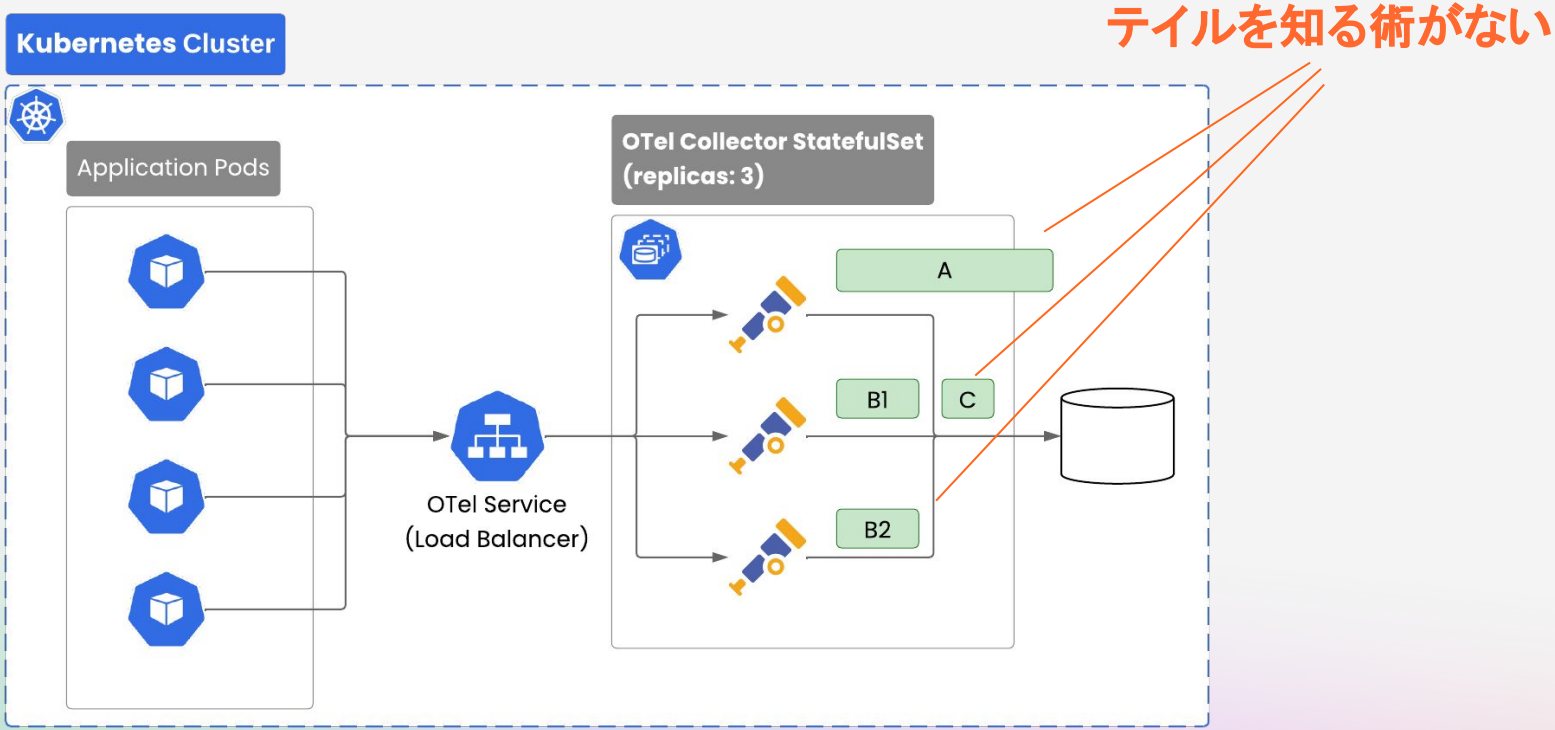
ルールベースサンプリング
- eg) ヘルスチェックの除外

テイルベースサンプリング

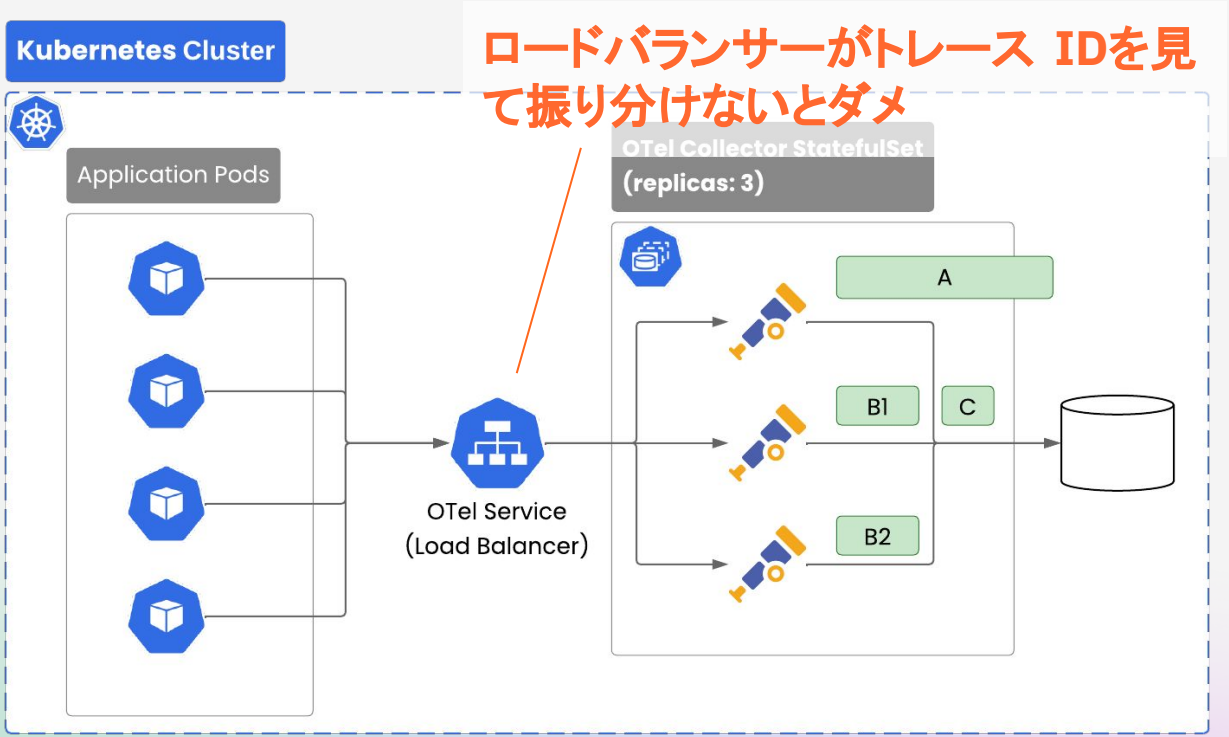
特定のトレースのすべてのスパンを受け取った後に記録するかを決定

ルールベースサンプリング
- eg) トレース全体のレイテンシー
- eg) スパン数の合計
- eg) 特定のスパンの有無

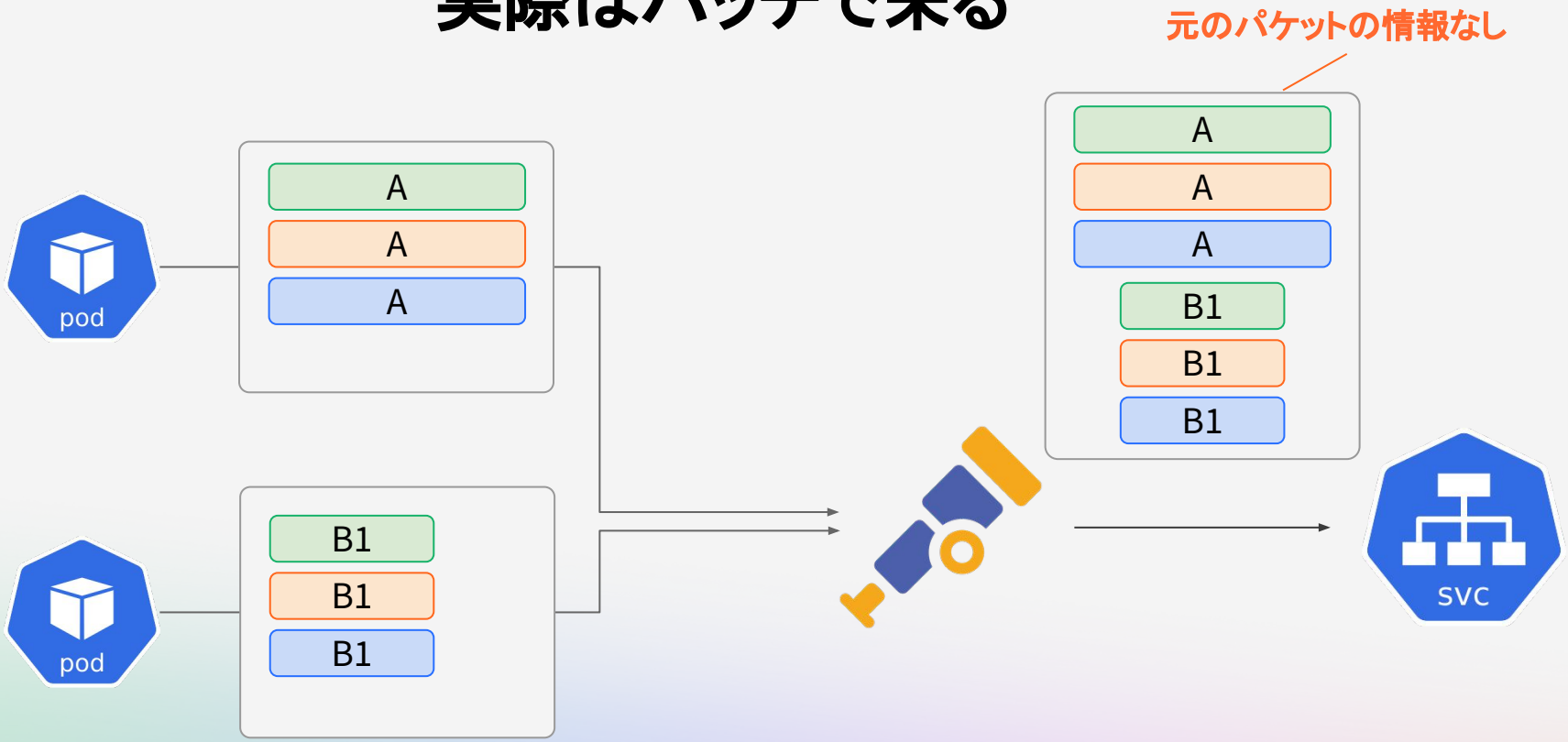
コレクタープール × テイルサンプリングの罠



コレクタープール × テイルベースサンプリングの罠



実際はバッチで来る



思っているより考えることが多い

課題

OTLPでロードバランサーに送信しているときに起きていること

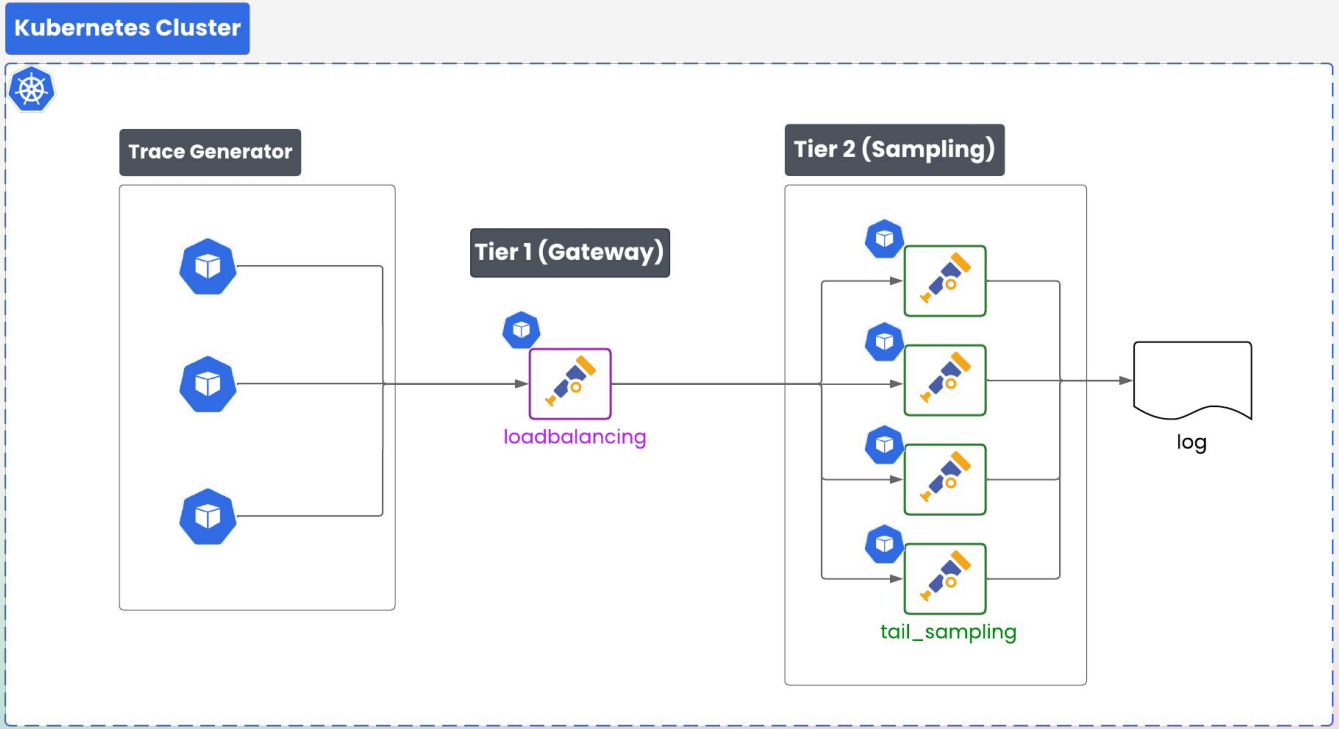
- エクスポーターからLBへのバッチ送信も1つのトレース
- 汎用ロードバランサーは送信トレースのヘッダーは見れる
- OTLP over gRPC は内部に複数のスパンデータを持つ

汎用L4/L7 LB だとバッチ自体のトレースしか見れない

オープンソースのみの構成

loadbalancing エクスポーター & tail_sampling プロセッサー

https://github.com/ymotongpoo/otel-lb-tailsampling-demo



loadbalancing エクスポーターの設定

```
exporters:
  loadbalancing:
    routing_key: "traceID"
    protocol:
      otlp:
    resolver:
      dns:
        hostname: otel-tier2-internal
        port: "4317"
```

トレースなら "service", "traceID", "attribute" が有効

コレクタープール側の設定

```yaml
kind: Service
metadata:
  name: otel-tier2-internal
spec:
  clusterIP: None # Headlessにする
  selector:
     app: otel-tier2
  ports:
     - port: 4317
     targetPort: 4317
```

Pod名を {StatefulSet名}-{通し番号} で固定

```yaml
kind: StatefulSet
metadata:
  name: otel-tier2
spec:
  serviceName: otel-tier2-internal
  replicas: 4
  selector:
     matchLabels:
     app: otel-tier2
  template:
     metadata:
     labels:
     app: otel-tier2
```

tail_samplingプロセッサーの設定

```
processors:
  batch:
  tail_sampling:
    decision_wait: 5s
    num_traces: 10000
    expected_new_traces_per_sec: 100
    policies:
      - name: latency-policy
        type: latency
        latency: {threshold_ms: 100}
```

思っているより考えることが多い

課題

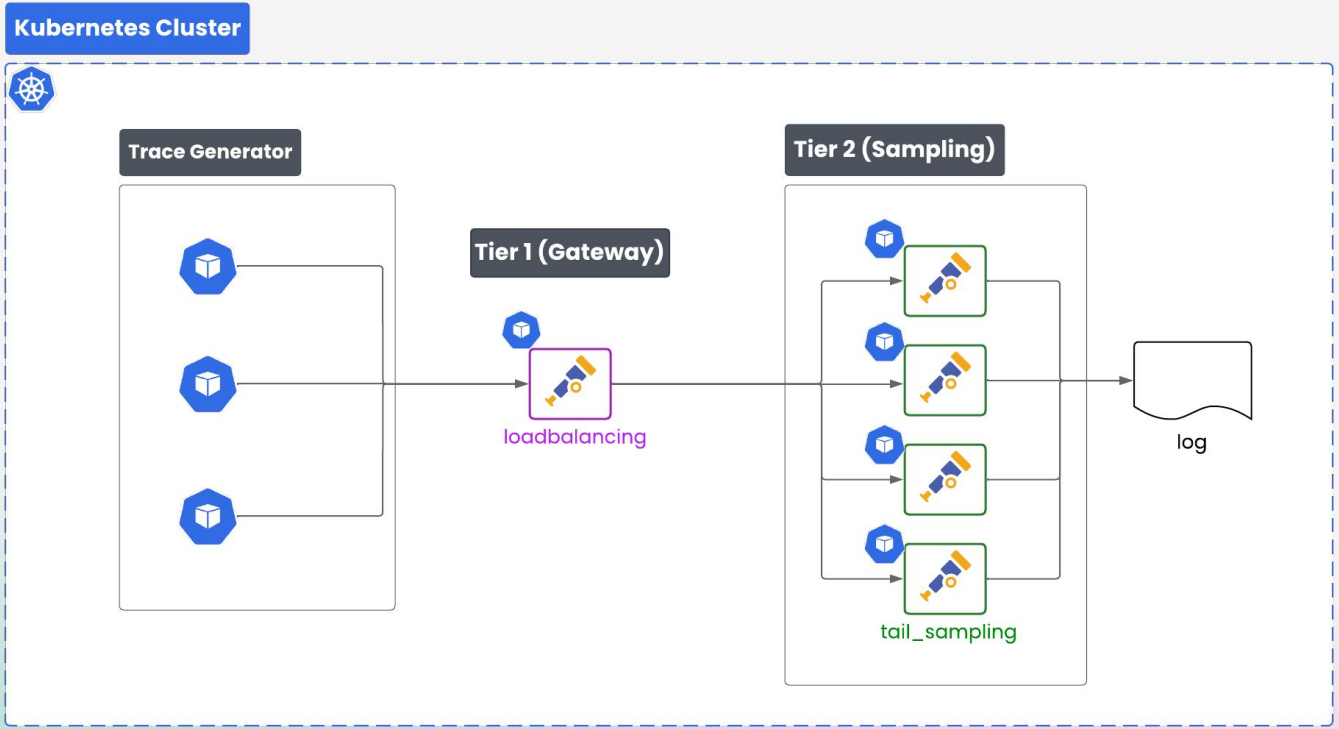
OTLPでロードバランサーに送信しているときに起きていること

- エクスポーターからLBへの送信も1つのトレース
- 汎用ロードバランサーは送信トレースのヘッダーは見れる
- OTLP over gRPC は内部に複数のスパンデータを持つ

loadbalancing エクスポーターならスパンレベルまで
protobuf の中身を見てルーティングできる

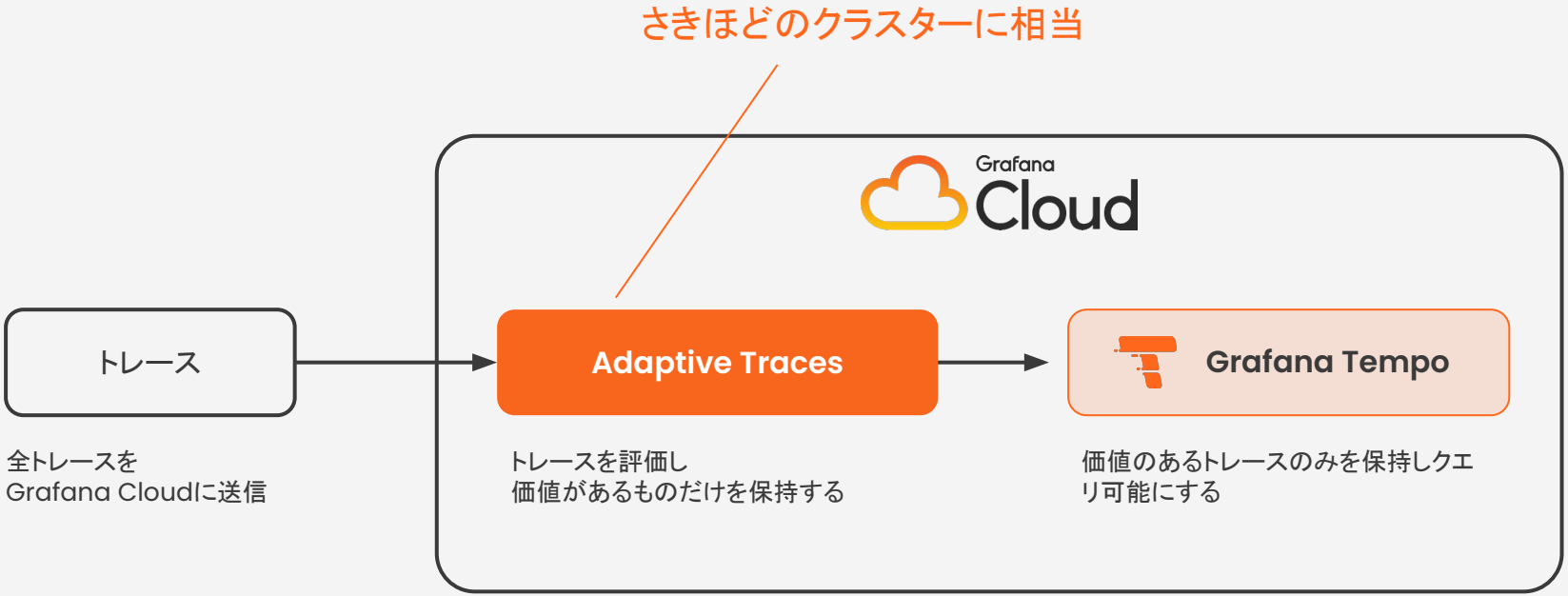
クラスターの運用はコスト

このサンプリング機構を本当に自分で運用したいのか



Grafana Cloudでの例

必要なところは外部サービスに任せる



まとめ

- テレメトリーのROIを高めるためにはサンプリングは1つの手段
- しかしテイルサンプリングを正しく行うのは難しい
- 運用コストとサービス利用コストを両方考えることが必要