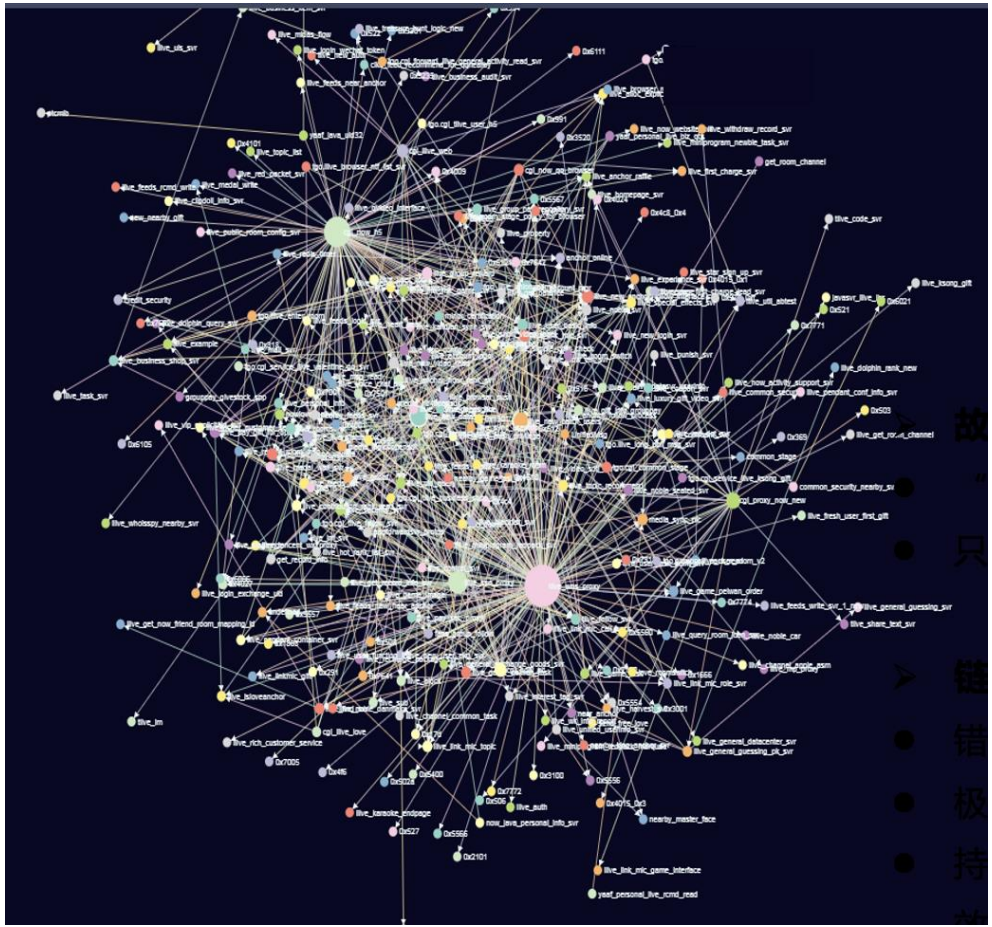
云原生时代分布式链路追踪实践

曲赛（saiqu）

2021-08

微服务架构的困境



故障定位难

日志分散

定位过程“击鼓传花”

性能分析难

跨端性能瓶颈分析繁杂

链路梳理难

极高的沟通和交接成本

错综难懂的模块依赖关系

架构治理能力匮乏

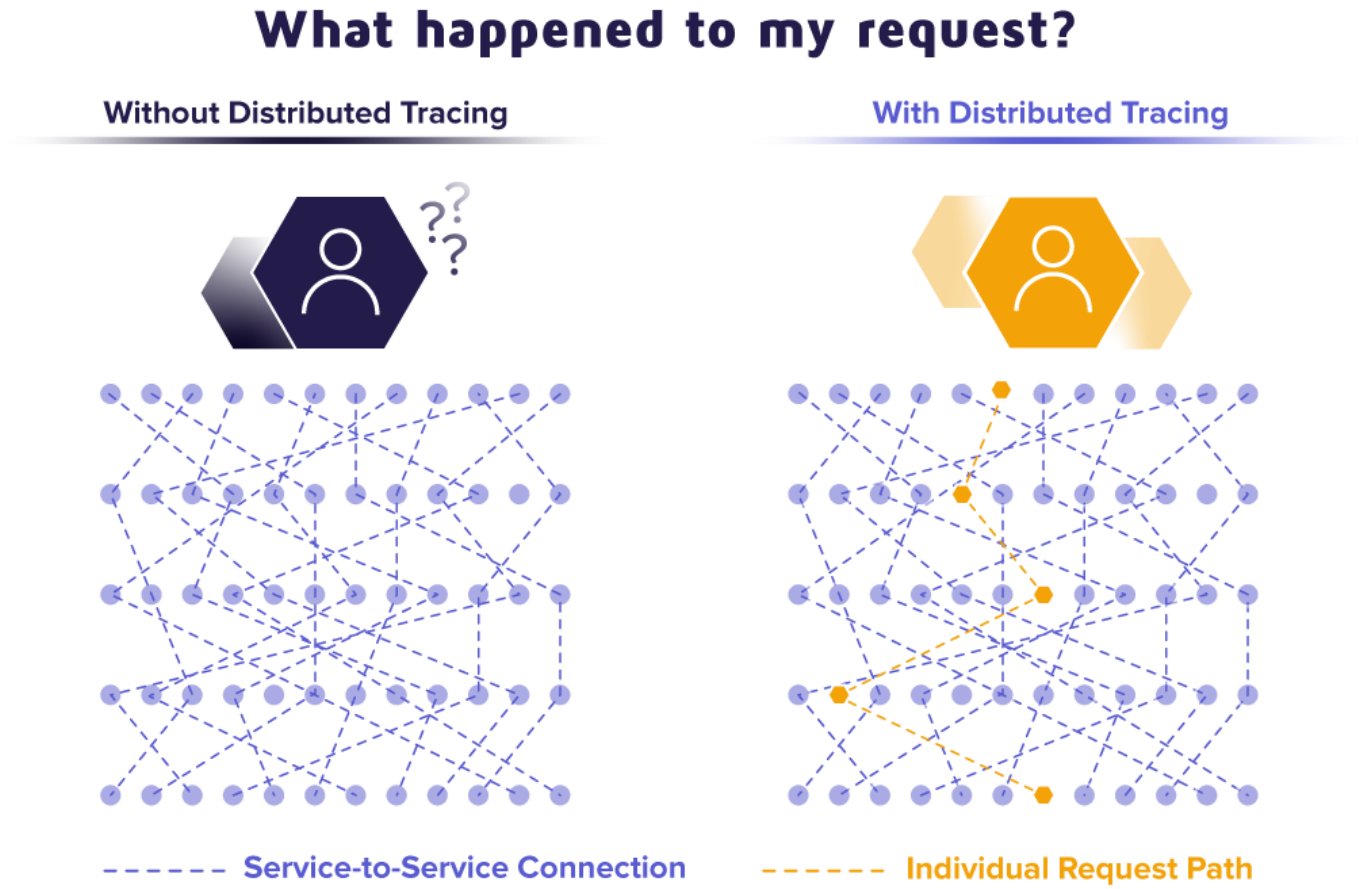
缺乏对系统整体认知的把控

不合理的调用关系

不合理的直连存储

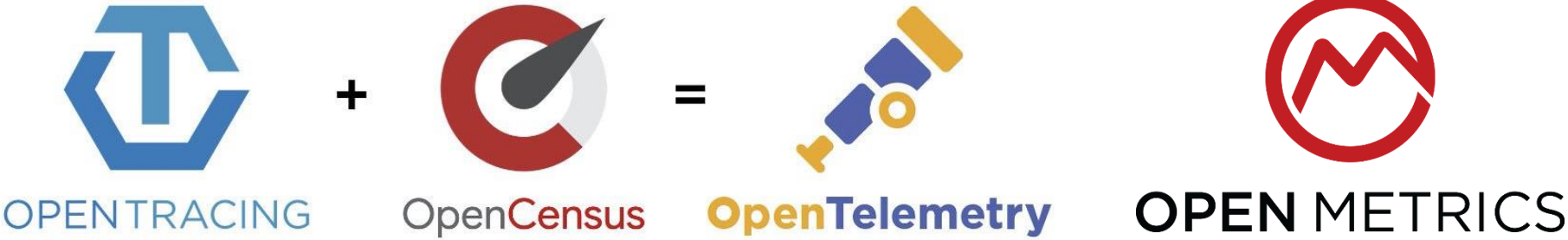
云原生可观测性

Trace


What happened to my request?
Without Distributed Tracing
With Distributed Tracing
Service-to-Service Connection
Individual Request Path

标准规范

标准	概述	Traces	Metrics	Logs	状态
OpenTracing	2015年底发起，2016年被批准为CNCF第三个项目	✓			停止更新
OpenCensus	2017年，起源于Google，项目负责人来自Google，Microsoft	✓	✓		停止更新
OpenMetrics	2017年，起源于Prometheus社区，项目负责人来自Grafana，Gitlab		✓		持续更新
OpenTelemetry	2019年，由OpenTracing和OpenCensus合并而来。	✓	✓	✓	蓬勃发展



Trace 数据模型：Trace Context，Baggage

Propagation Format
W3C Trace-Context
W3C Baggage
Zipkin B3 format
Jaeger
AWS X-Ray

请求

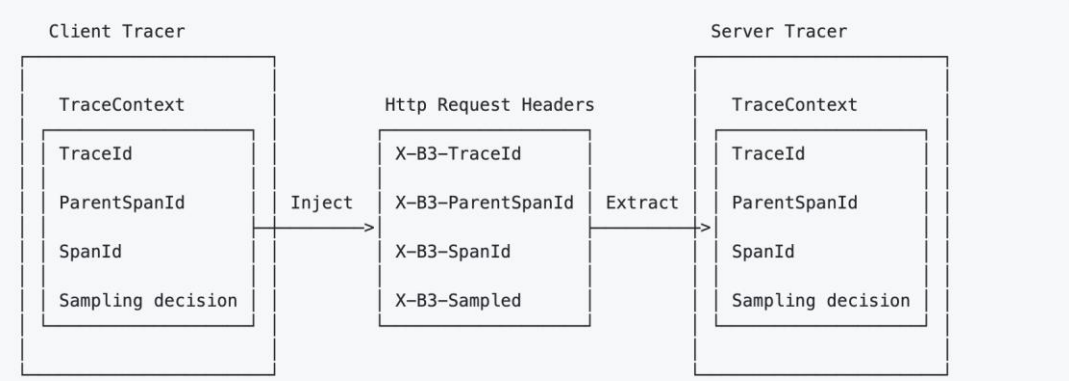
traceparent: 00-4bf92f3577b34da6a3ce929d0e0e4736-d75597dee50b0cac-01

tracestate: rojo=00f067aa0ba902b7,congo=t61rcWkgMzE

baggage: userId=alice,serverNode=DF:28,isProduction=false

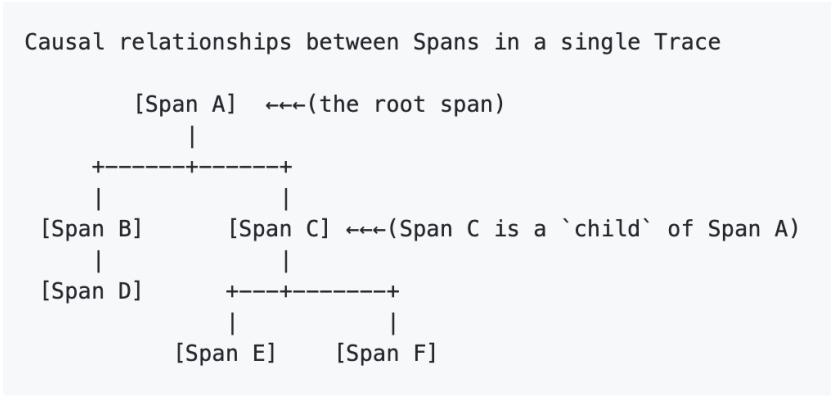
响应

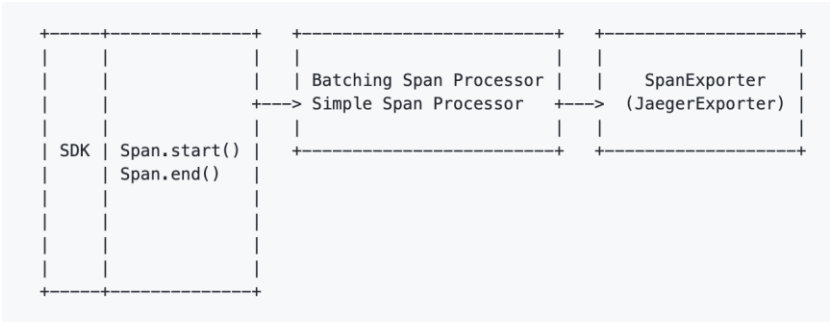
traceresponse: 00-1baad25c36c11c1e7fbd6d122bd85db6-



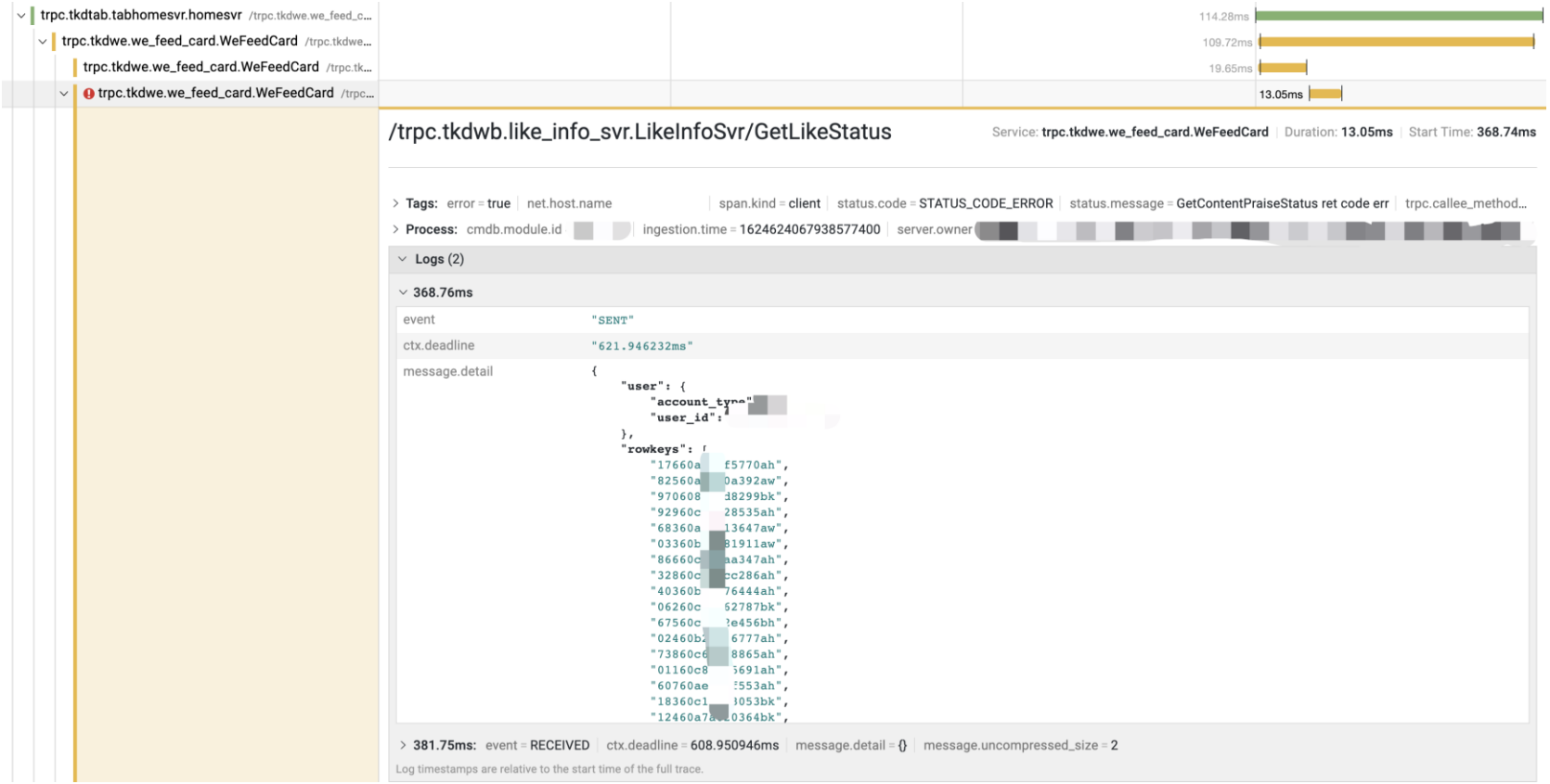
Trace 数据模型： Trace Detail

```
message Span {
  bytes trace_id = 1;
  bytes span_id = 2;
  string trace_state = 3;
  bytes parent_span_id = 4;
  string name = 5;
  SpanKind kind = 6;
  fixed64 start_time_unix_nano = 7;
  fixed64 end_time_unix_nano = 8;
  // 属性键值对
  repeated KeyValue attributes = 9;
  // 内嵌日志(事件)
  repeated Event events = 11;
  // 链接到其它Trace
  repeated Link links = 13;
  Status status = 15;
}
```

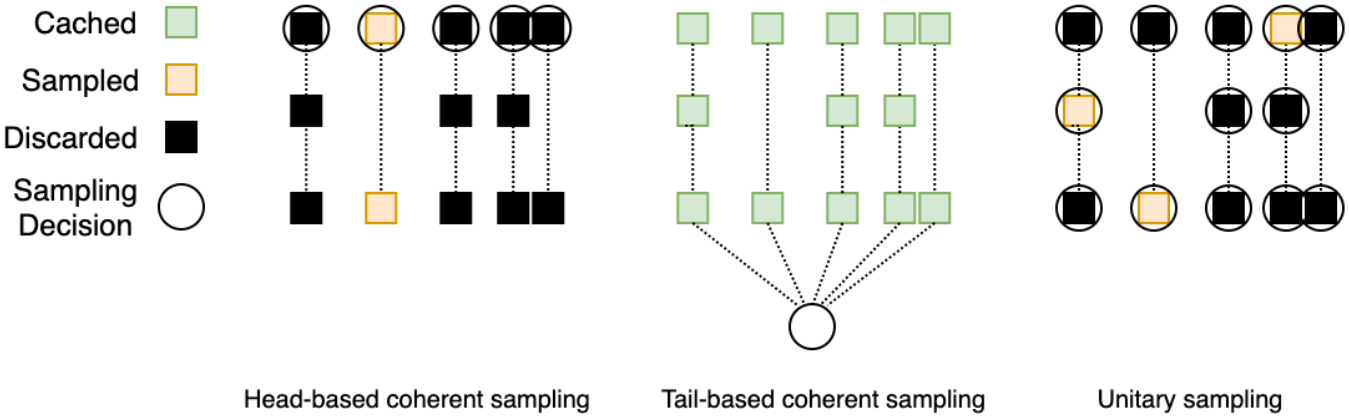




Trace 数据模型： Trace Detail 示例

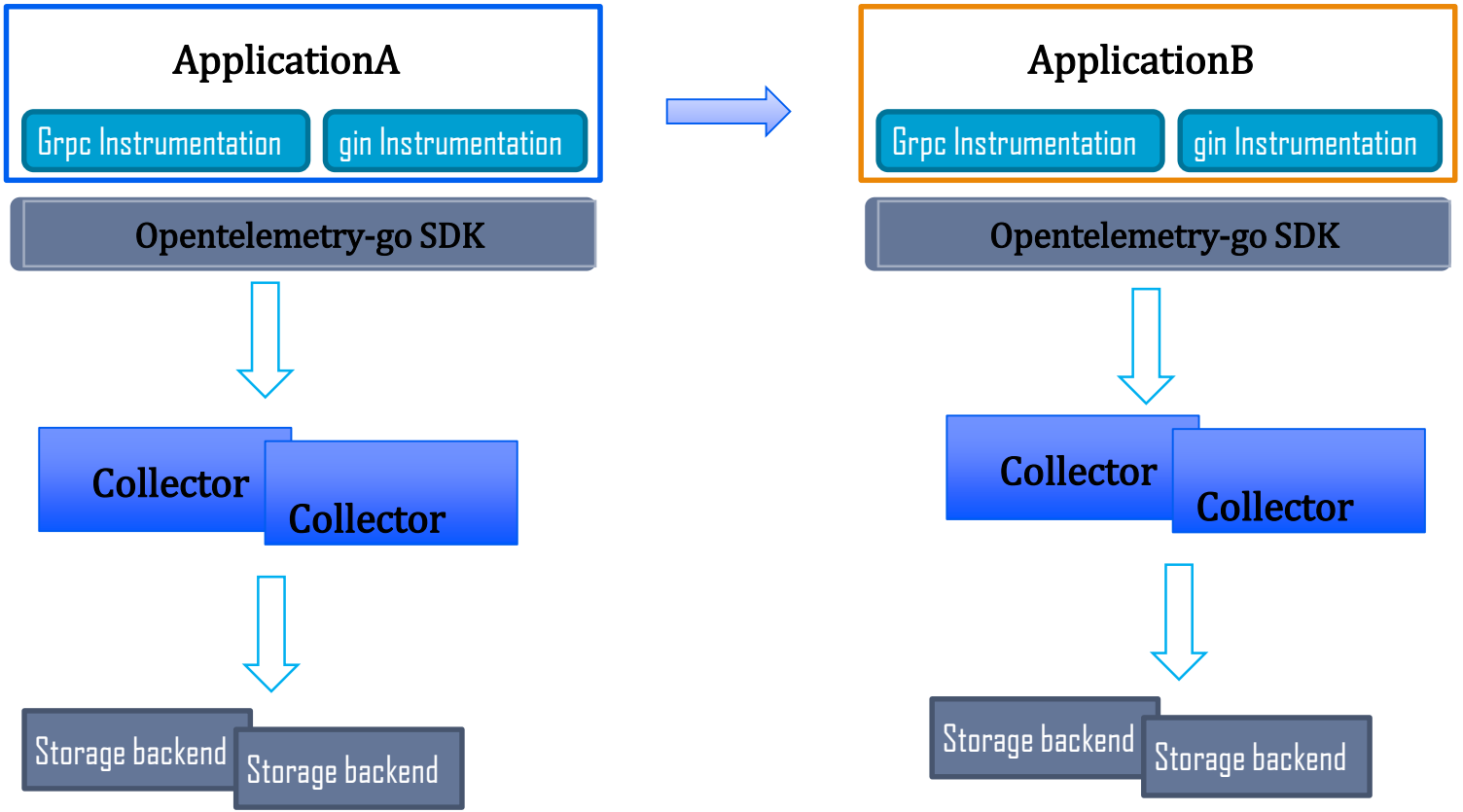


Trace 采样策略



1. Head-based coherent sampling
2. Tail-based coherent sampling
3. Unitary sampling
4. 多维度染色采样：指定某用户或指定某文章采样

Trace 传递与采集

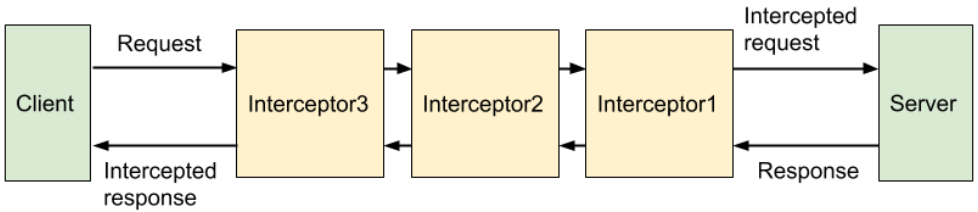

ApplicationA
Grpc Instrumentation
gin Instrumentation
Opentelemetry-go SDK
Collector
Collector
Storage backend
Storage backend
ApplicationB
Grpc Instrumentation
gin Instrumentation
Opentelemetry-go SDK
Collector
Collector
Storage backend
Storage backend

Instrumentatition 非侵入式的业务接入

利用拦截器机制的实现

接入便利，只需引入对应的拦截器组件

trace基础属性自动采集



一次网络调用的经过的拦截器数据流

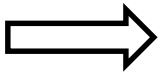
```
name, attr := spanInfo(method, cc.Target())
var span trace.Span
ctx, span = tracer.Start(
        ctx,
        name,
        trace.WithSpanKind(trace.SpanKindClient),
        trace.WithAttributes(attr...),
)
defer span.End()

Inject(ctx, &metadataCopy, opts...)
ctx = metadata.NewOutgoingContext(ctx, metadataCopy)

messageSent.Event(ctx, 1, req)

err := invoker(ctx, method, req, reply, cc, callOpts...)

messageReceived.Event(ctx, 1, reply)

if err != nil {
        s, _ := status.FromError(err)
        span.SetStatus(codes.Error, s.Message())
        span.SetAttributes(statusCodeAttr(s.Code()))
} else {
        span.SetAttributes(statusCodeAttr(grpc_codes.OK))
}
```

otelgrpc instrumentation核心实现

天机阁2.0 简介

天机阁2.0是遵循OpenTelemetry标准的，为各业务或平台提供分布式追踪，监控，日志，多维染色，容量评估，架构治理等能力的云原生可观测性系统。

- 分布式追踪
- 日志
- 服务监控
- 火焰图
- 存储监控
- SDK监控
- CI/CD监控
- 发布变更
- 告警历史
- 服务拓扑图



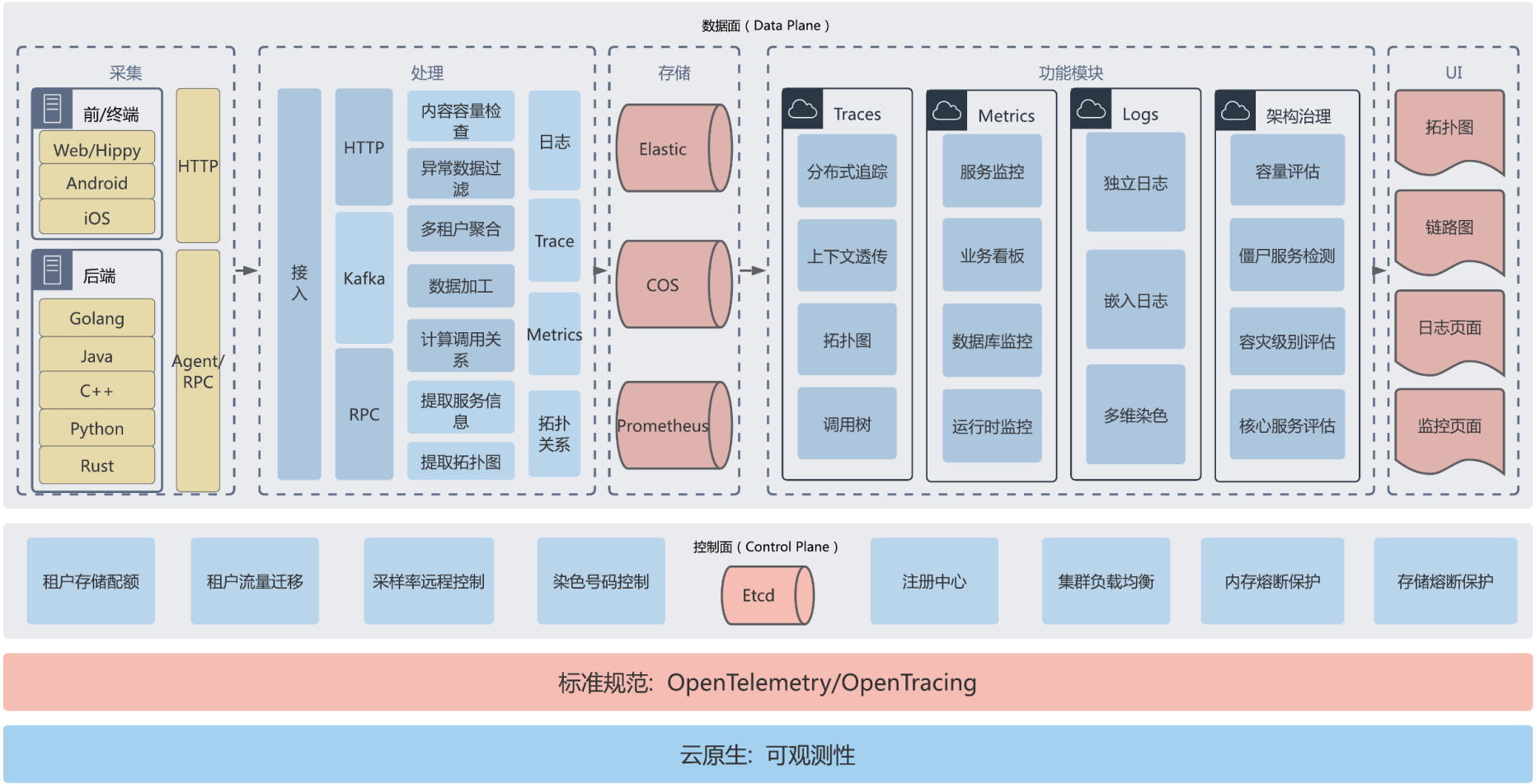
愿景：让开发一切尽在掌握

正交，模块化

多租户

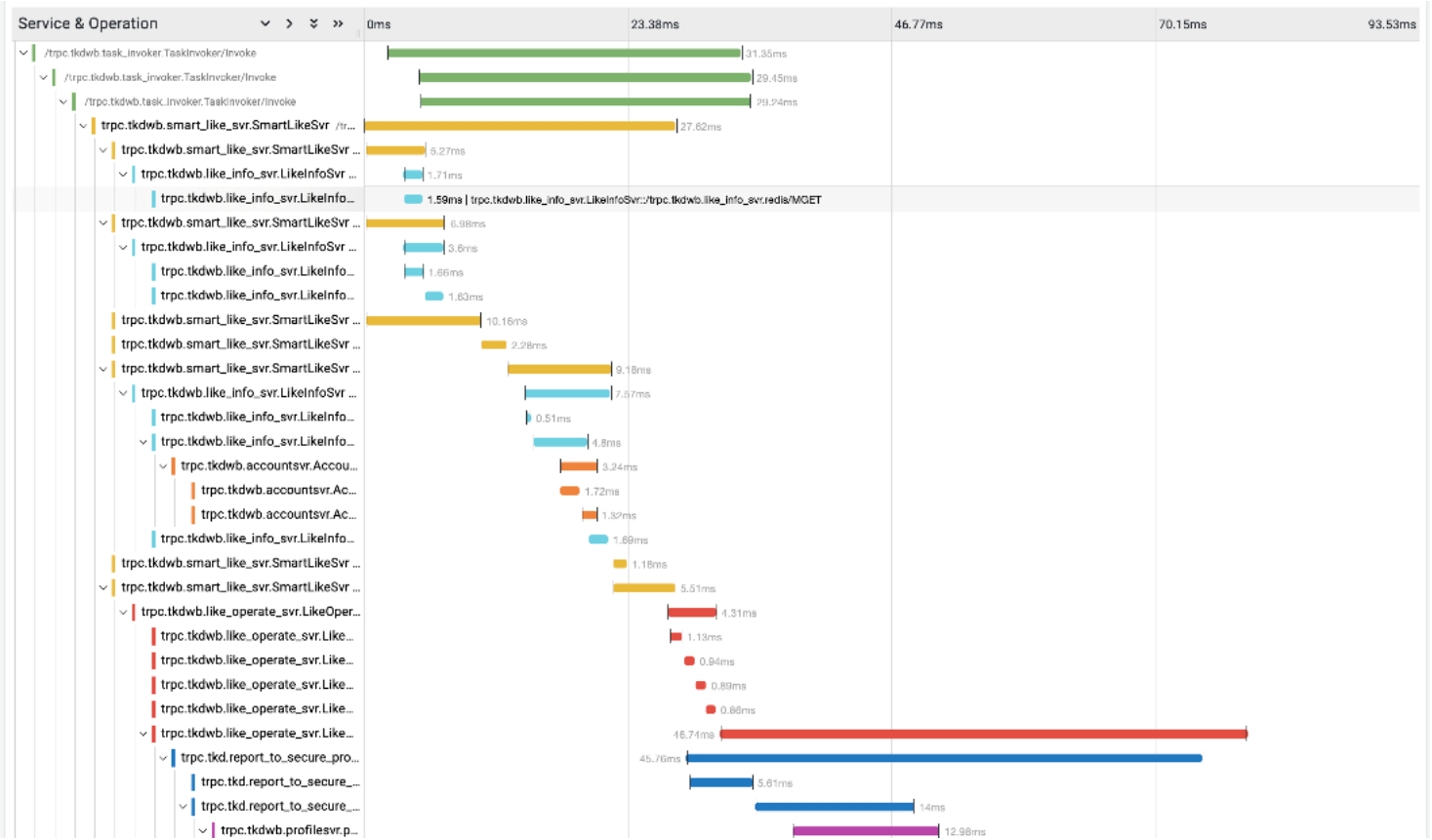
相关性

天机阁2.0 架构


数据面（Data Plane）
采集
前/终端
Web/Hippy
Android
iOS
后端
Golang
Java
C++
Python
Rust
HTTP
Agent/
RPC
处理
接入
HTTP
Kafka
RPC
内容容量检查
异常数据过滤
多租户聚合
数据加工
计算调用关系
提取服务信息
提取拓扑图
日志
Trace
Metrics
拓扑关系
存储
Elastic
COS
Prometheus
功能模块
Traces
分布式追踪
上下文透传
拓扑图
调用树
Metrics
服务监控
业务看板
数据库监控
运行时监控
Logs
独立日志
嵌入日志
多维染色
架构治理
容量评估
僵尸服务检测
容灾级别评估
核心服务评估
UI
拓扑图
链路图
日志页面
监控页面
控制面（Control Plane）
租户存储配额
租户流量迁移
采样率远程控制
染色号码控制
Etcd
注册中心
集群负载均衡
内存熔断保护
存储熔断保护
标准规范：OpenTelemetry/OpenTracing
云原生：可观测性

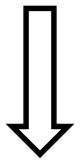
天机阁2.0 实践

分布式追踪



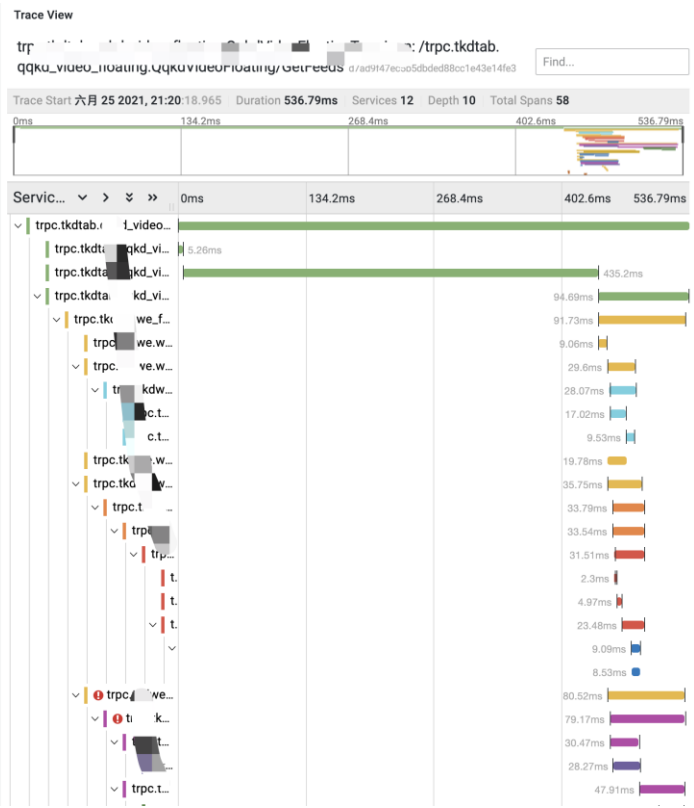
天机阁2.0 实践

Log详情



分布式追踪

点击Log详情中traceID字段的按钮拉起Trace详情。



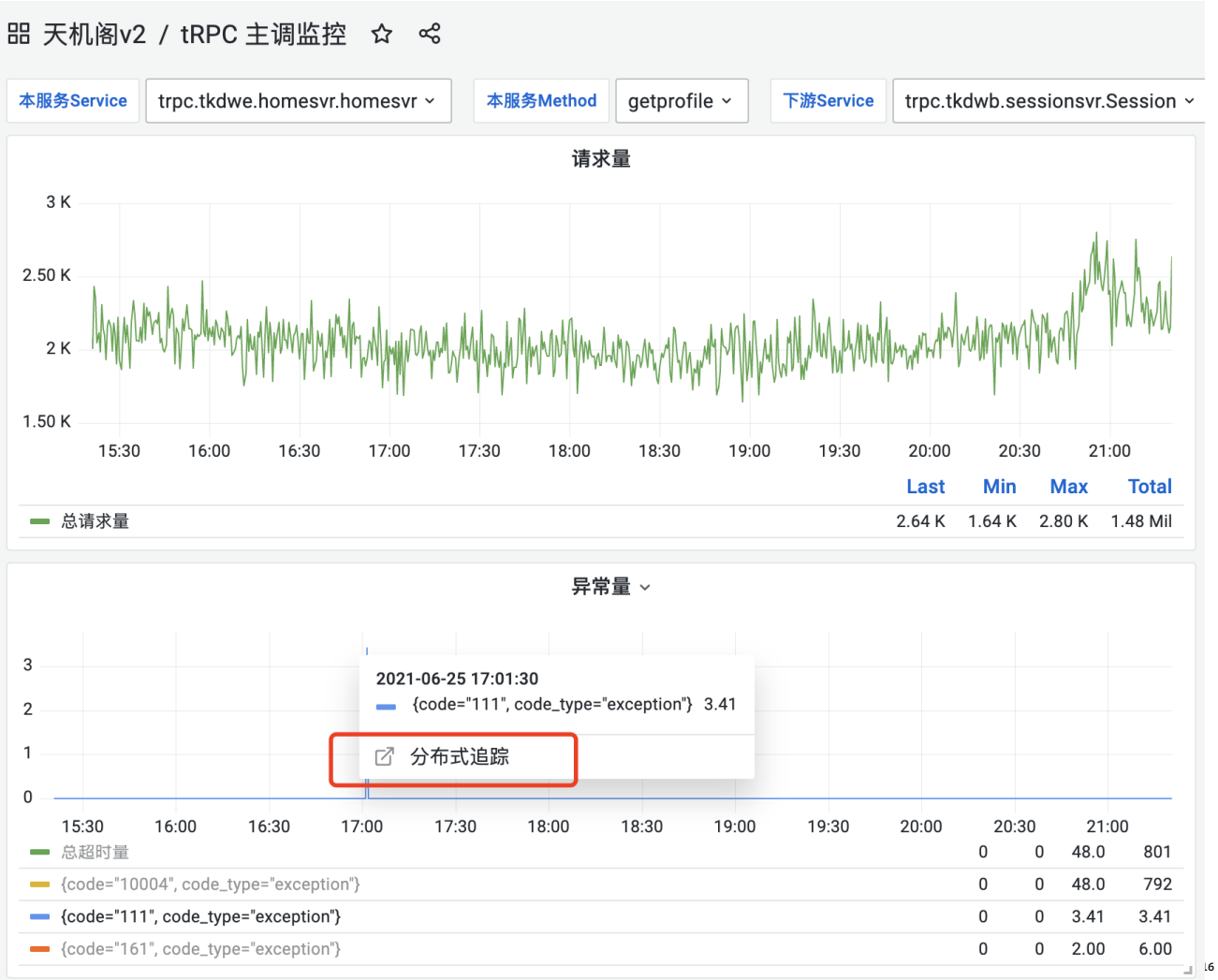
天机阁2.0 实践

监控面板

⇩

分布式追踪

监控到错误码111，点击面板跳转到相关时间段的分布式追踪



感谢倾听