Distributed tracing – maximizing ROI for OpenTelemetry integration

分布式跟踪 – 最大限度地提高 OpenTelemetry 集成的投资回报率

Ilya Mochalov

Content

Part 01
Background

Why latency matters? 为什么延迟很重要？

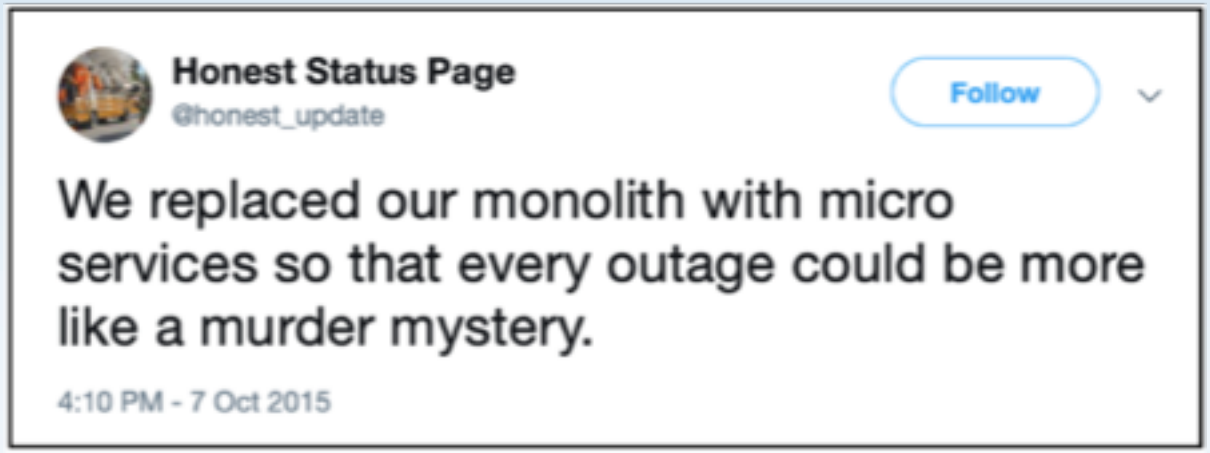
LATENCY = DELAY = 延迟

- Google:
  - **increasing the time to load a page of search results by half a second reduced the total number of searches by 20%**
  - and conversely, if Google made a page faster, it would see a roughly **proportional increase in the amount of usage**.
- Akamai
  - latency of **100 milliseconds could reduce ecommerce conversion by as much as 7%**.
- Pinterest
  - **40% reduction in visitor wait time resulted in a 15% increase** (sign-up conversion).

Source: Chapter 8 of Distributed Tracing in Practice - Instrumenting, Analyzing, and Debugging Microservices (O'Reilly Media)

Why distributed tracing? 为什么要进行分布式跟踪？

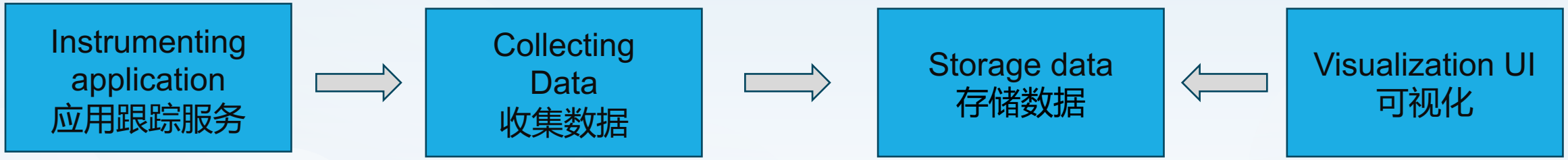
Challenge:



Traditional logs and metrics 传统日志和指标	Distributed tracing 分布式跟踪
• lack of context 缺乏背景 • needs manual correlation 需要手动关联 • easy to create too verbose (logs) 容易创建太冗长（日志） • cardinality issue for metrics with many dimensions 具有多个维度的指标的基数问题	• show whole request 显示整个请求 • event centric 以事件为中心 • contextual metadata 上下文元数据 • relationships between different services 不同服务之间的关系

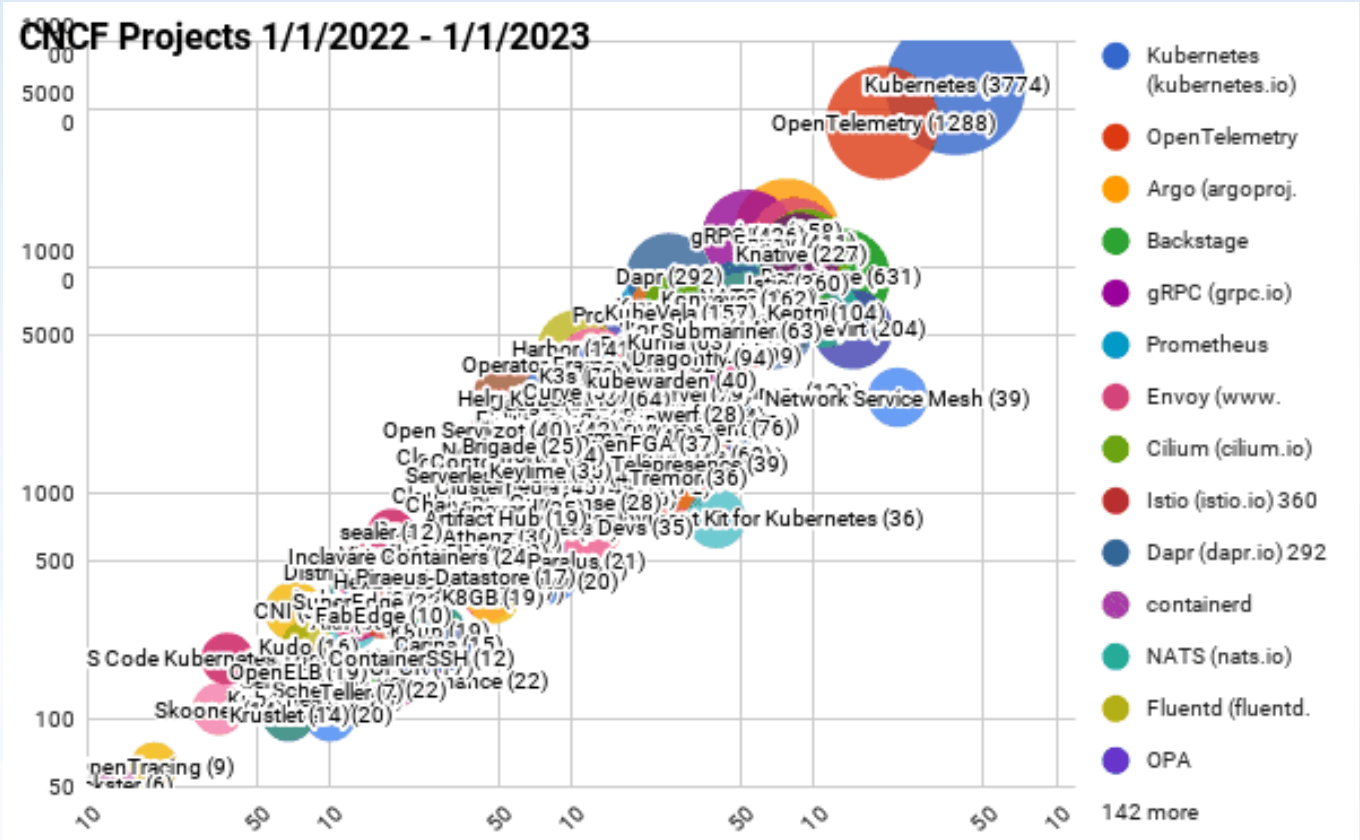
Distributed tracing system 分布式跟踪系统

Any distributed tracing system consists of the following parts:



Why OpenTelemetry?

- Vendor- and tool-agnostic 与供应商和工具无关
- Single set of APIs and conventions 通用的 API 和约定
- Helps with instrumentation 帮助增加跟踪
- Colleting metrics for sampling and transformation 用于采样和转换的指标收集
- Integrate OpenTelemetry **once** and use it with **any** storage and UI 只需集成一次 OpenTelemetry，即可将其用于任何存储和 UI



OpenTelemetry is 2nd fastest growing project
OpenTelemetry 是增长速度第二的项目
in https://cncf.io/ landscape in 2022

Part 02

Methodology for maximizing ROI
最大化投资回报率的实现方法论

Methodology 方法论

Follow 5 steps process when working with distributed tracing:

Avoiding common mistakes 避免常见错误

- Not colleting traces into a single backend 不将跟踪整理到一台数据库
- Not enforcing standards 不执行标准
- Multiple teams working at the same time 多个团队同时工作
- Not building internally used blueprints, libraries, documentation and best practices 不构建内部使用的蓝图、库、文档和最佳实践

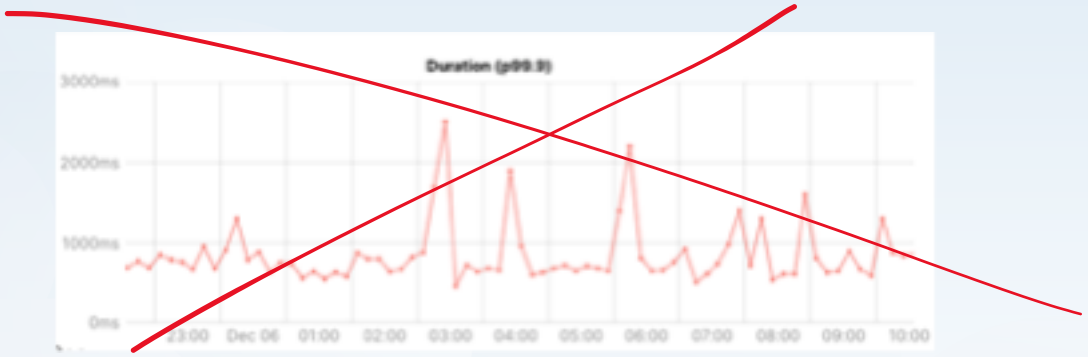
Part 03

Getting value from distributed traces

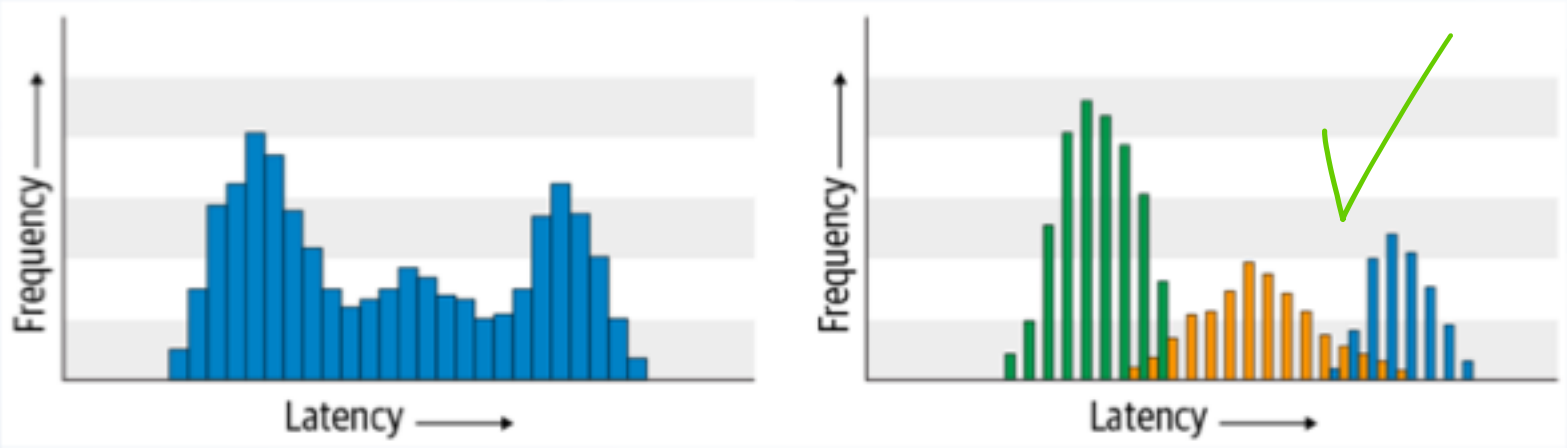
从分布式跟踪中获取价值

Find statistically significant requests 查找具有统计意义的请求

Use **histograms** instead of **time-series charts** 使用**直方图**而不是**时间序列图表**



Time-series charts shows **average** performance for your application 时间序列图表显示应用程序的**平均性能**

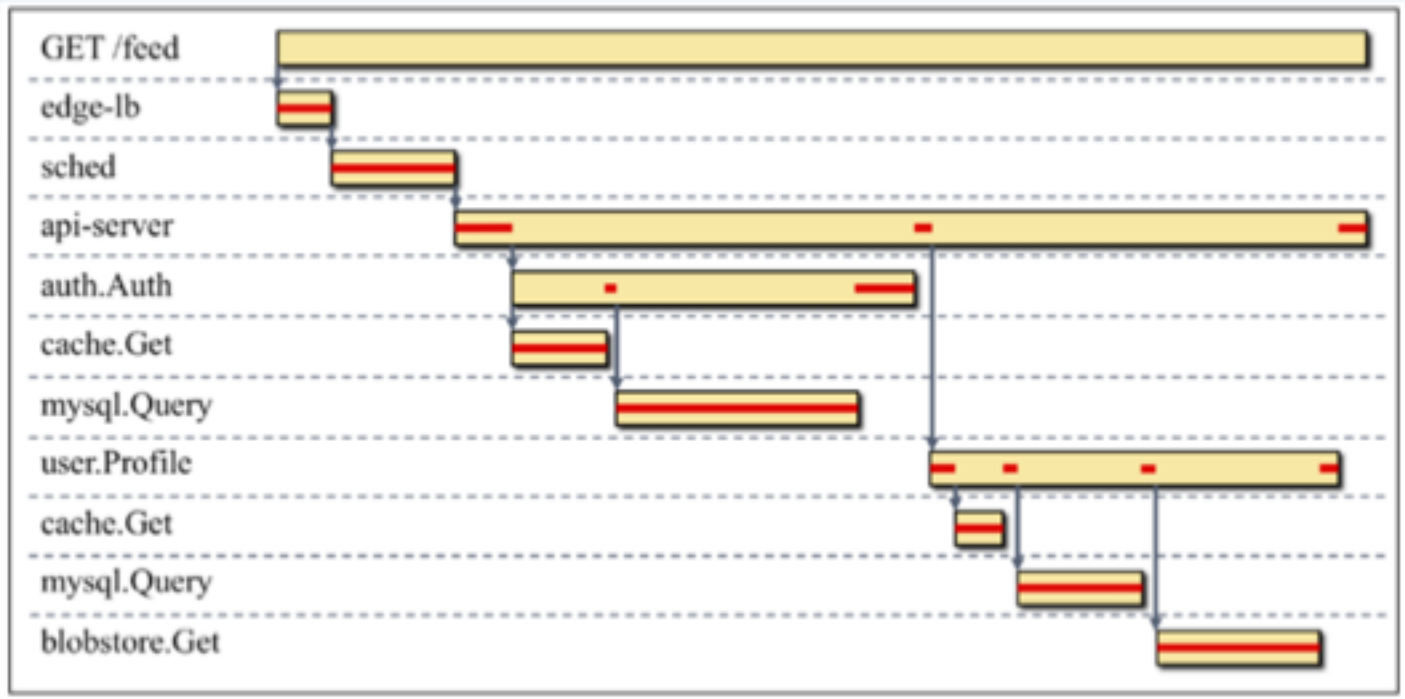


Histograms can show how many **normal distributions** you have in a dataset. It helps to identify **groups** of requests

直方图可以显示数据集中有多少个正态分布。它有助于识别请求组

Review traces – Critical path 关键路径

The critical path is combination of parts that account for overall end-to-end latency. 关键路径是考虑整体端到端延迟部分的组合。

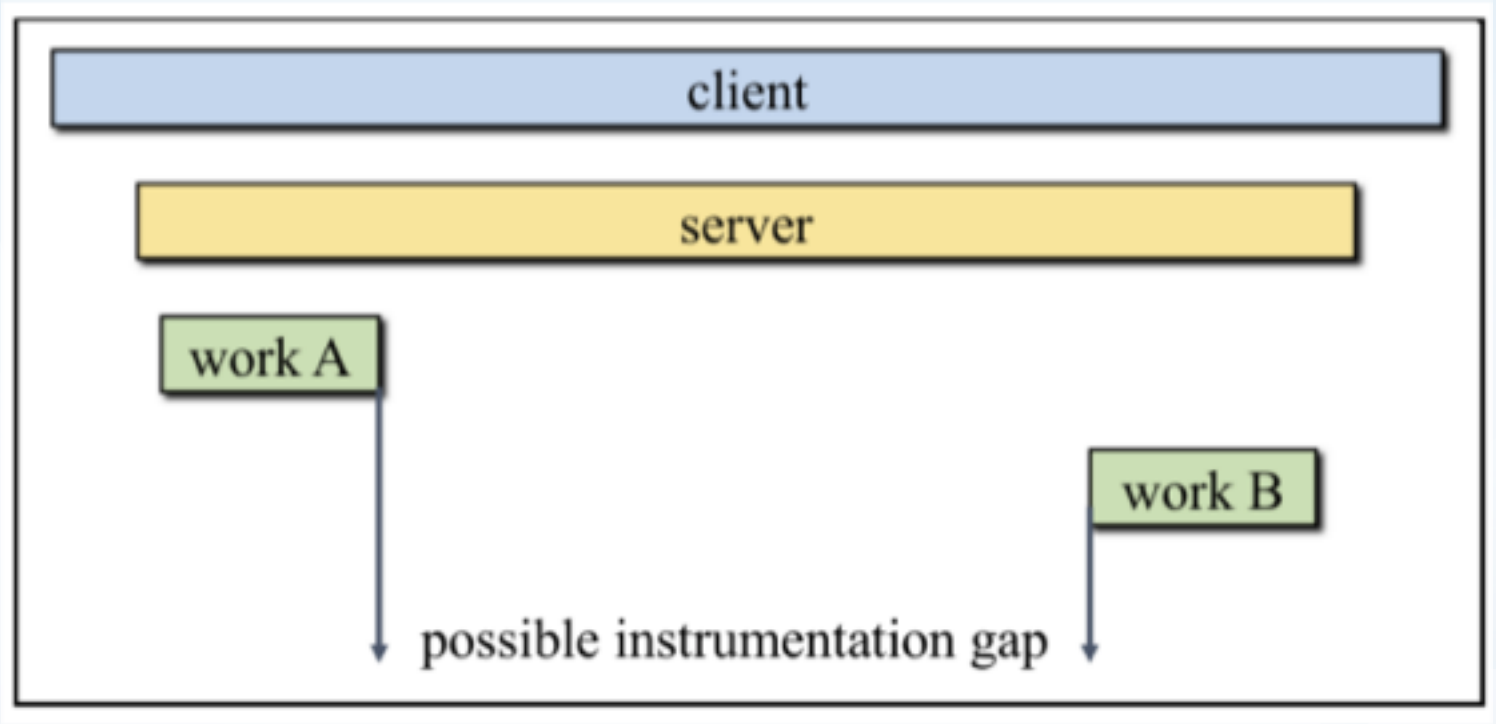


Benefits:

- we can **ignore** all spans that are off the critical path (optimizing them will not reduce latency) 我们可以**忽略**所有偏离关键路径的跨度（优化它们不会减少延迟）
- look for a longest span to find **most impactful parts to optimize** (20/80) 寻找最长的跨度以找到**最有影响力的部分**进行优化（20/80）

Review traces – Missing spans 缺少跨度

Missing spans (big gaps between spans) show missing instrumentation or a failure in tracing system
缺少跨度（跨度之间的间隙很大）表示缺少跟踪或跟踪系统出现故障

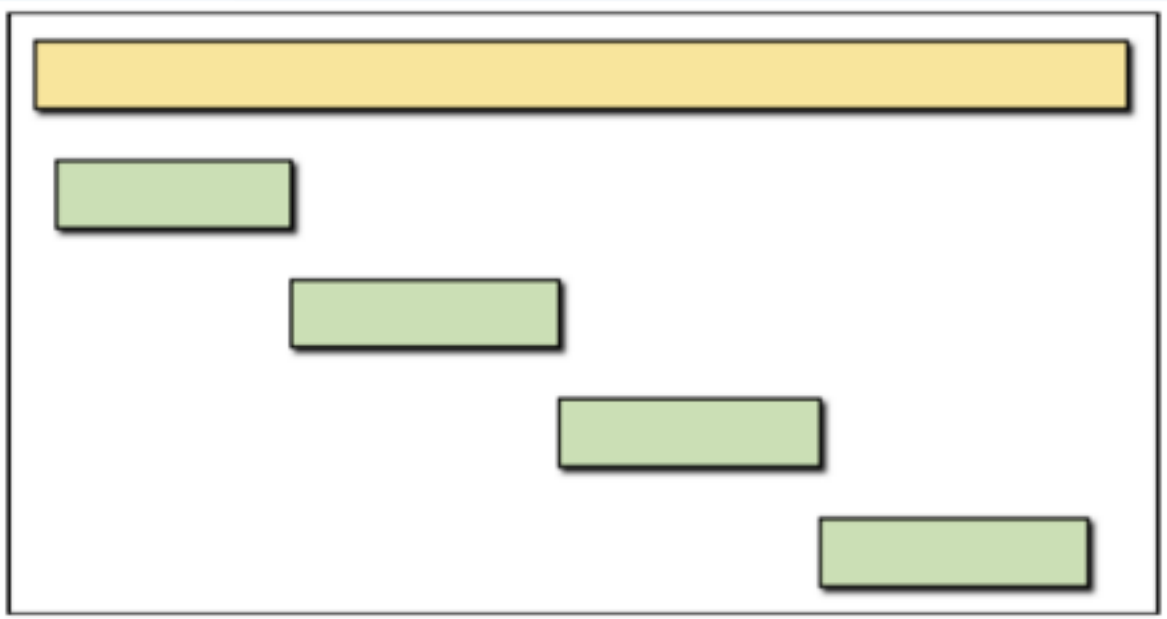


Example of a trace with missing spans

Review traces – Sequential execution 顺序执行

Sequential execution or "staircase" - not always an issue, but we need to review it. In most cases it is possible to refactor with **parallel** execution

顺序执行或“阶梯”——并不总是有问题，但我们需要审查它。在大多数情况下，可以通过并行执行进行重构

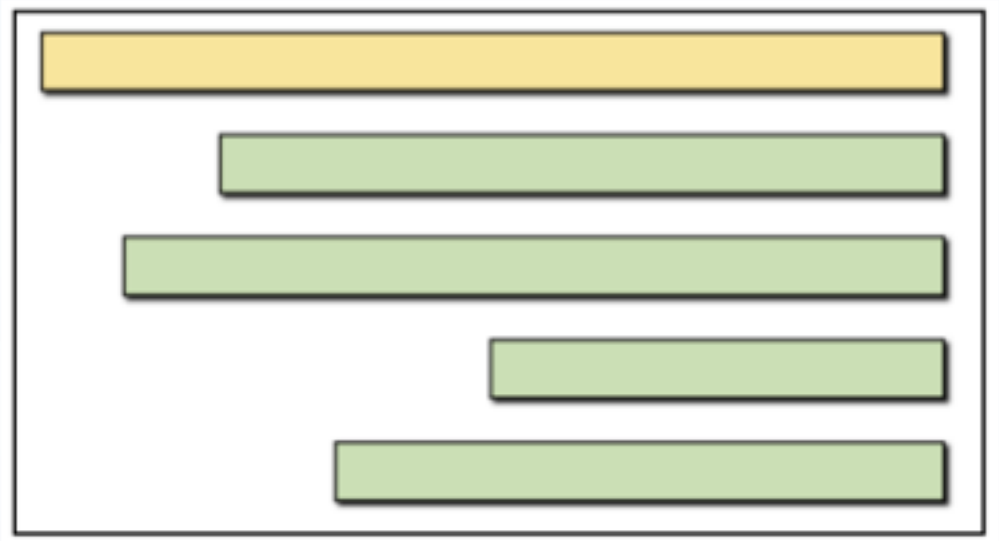


Example of sequential execution transaction

Review traces – Spans finish at the same time 跨度同时完成

It happens when:

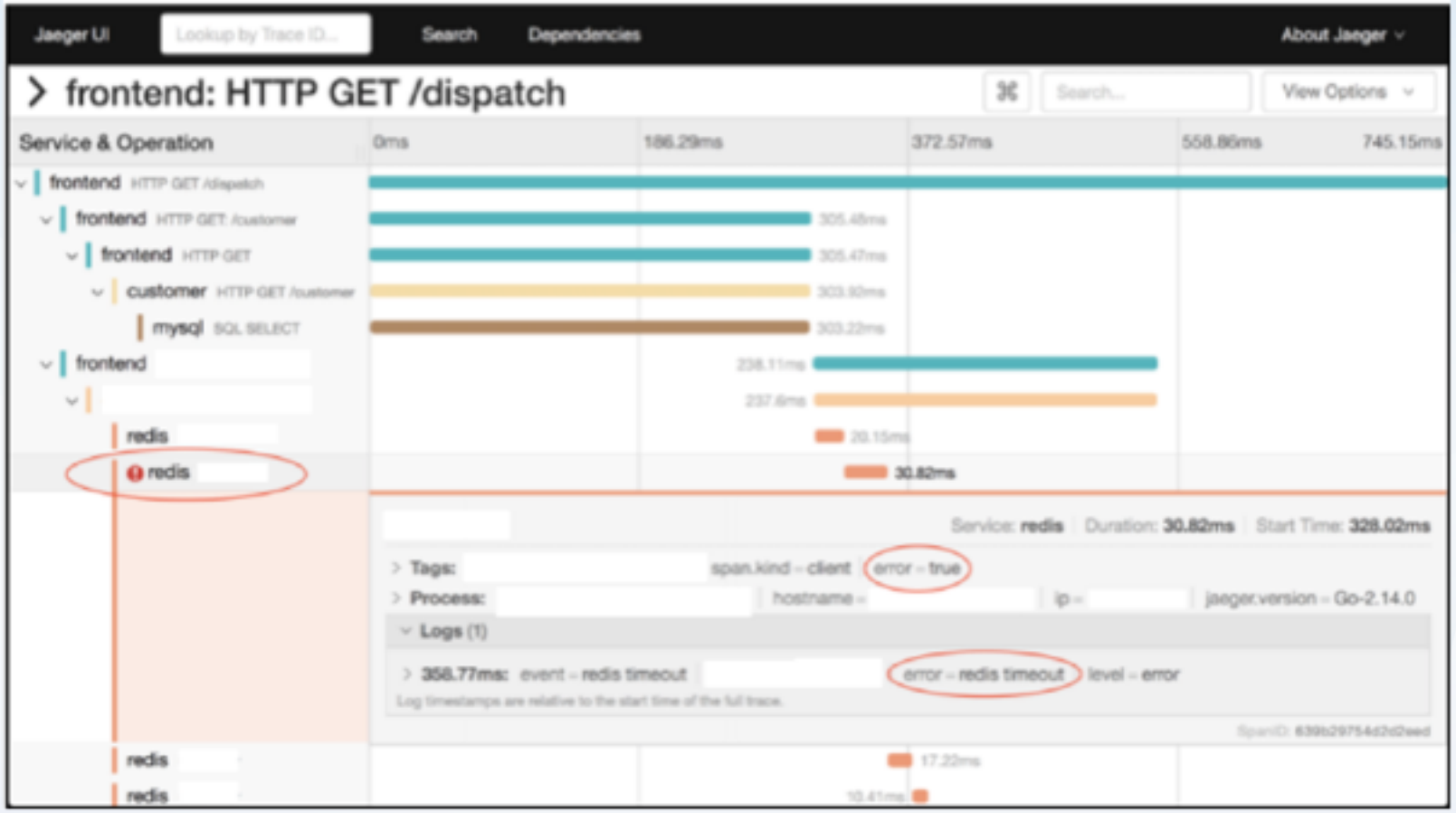
- the system supports timeouts, some tasks can wait for other tasks to finish, and it all get canceled altogether when timeout is reached 系统支持超时，部分任务可以等待其他任务完成，达到超时后全部取消
- requests are waiting on some kind of lock (long-running database transaction for example) 请求正在等待某种锁（例如，长时间运行的数据库事务）



Example of a request with spans finishing at the same time

Review traces – Failed spans 失败的跨度

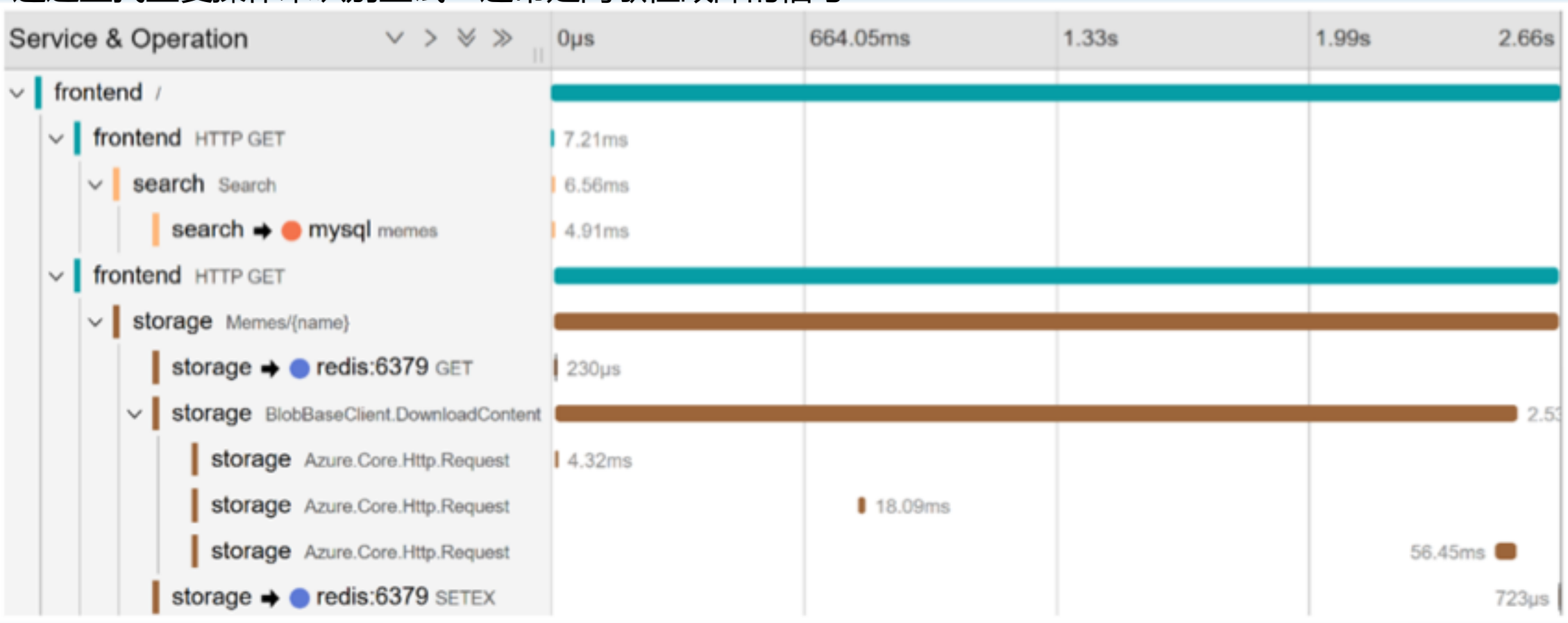
Make sure to check all spans marked as failed. Check for span attributes – you can find error messages 确保检查所有标记为失败的跨度。检查 span 属性 – 您可以找到错误消息



Example of failed span within a trace

Review traces – Retries 重试

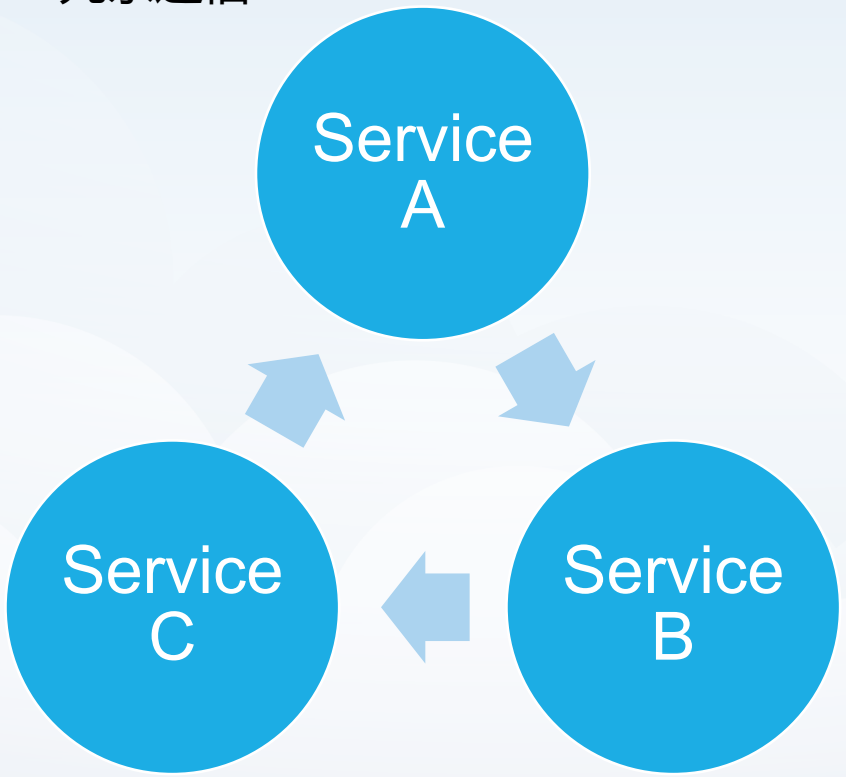
Retries can be identified by finding repetitive operations – usually signal of intermittent failures 可以通过查找重复操作来识别重试 - 通常是间歇性故障的信号



Example of a retried spans. A call to Azure storage had 3 attempts with back-off like time intervals in between. First 2 calls were canceled or gracefully failed

Review traces – Architectural problems 架构问题

- circular communications and inter-dependencies (two or more services depend on each other)
  循环通信和相互依赖（两个或多个服务相互依赖）
- redundant communications 冗余通信

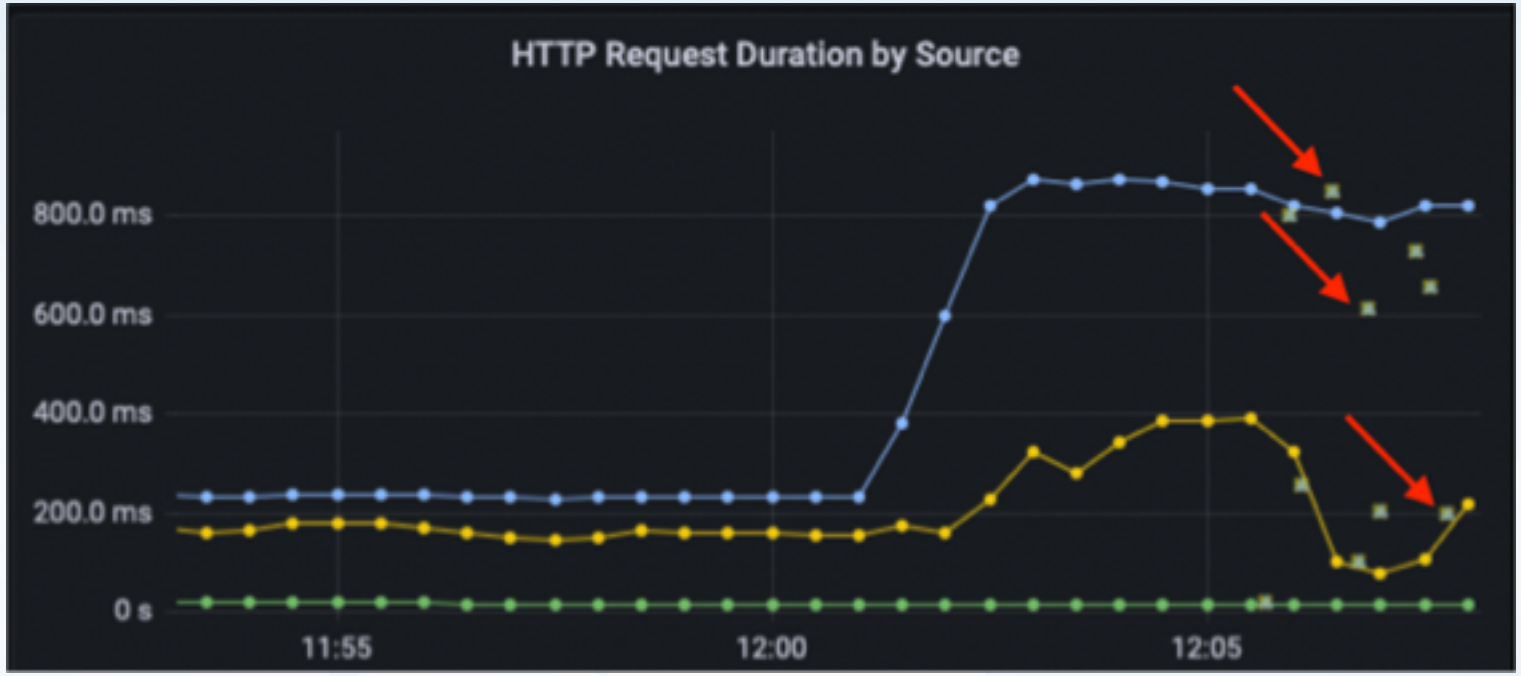


Example diagram is circular communication between services

Part 04
Next steps

Integrating with logs and metrics 与日志和指标集成

Metrics, logs, and traces are the "three pillars of observability". Integrating these 3 data types helps to **relate** events between them 指标、日志和跟踪是“可观测性的三大支柱”。集成这 3 种数据类型有助于在它们之间关联事件



Example of metrics and traces integration. Dota are exemplars (examples) of traces observed and related to the metrics

Profiling 应用分析

Distributed traces help you to identify places where latency happens, but in most cases, it is not obvious what causes it. To get more insights on application execution please do profiling. 分布式跟踪可帮助您确定发生延迟的位置，但在大多数情况下，导致延迟的原因并不明显。要获得有关应用程序执行的更多见解，请进行应用分析。

Example profiling flame chart

Recap

01 Why distributed tracing with Open Telemetry?

02 Iterative mindset + avoid common mistakes

03 Getting value

04 Corelating data + profiling

Thanks !