Наблюдаемость систем и процессов

Бочаров Филипп

О себе

Бочаров Филипп

Руководитель проектов по разработке в МТС ИТ

Занимаюсь разработкой платформы Наблюдаемости. Помогаю продуктовым командам сделать работу сложных распределенных систем понятной и прозрачной.

Цели доклада

1. мотивировать заниматься наблюдаемостью,
2. сэкономить ваше время,
3. дать представление об “идеальной” картине наблюдаемости.

НЕ цели: рекламировать конкретные инструменты или решения.

Наблюдаемость - возможность задавать вопросы о работе системы.

Проблемы экосистем на примере МТС

Экосистема МТС

- **400+ цифровых продуктов**: телемедицина, Smart Farming, Big Data, умный дом, внутренние платформы …
- **Гетерогенность IT-ландшафта**: продукты используют разные языки, платформы и фреймворки
- **Экосистема продуктов:** продукты взаимодействуют между собой, увеличивая ценность предложения

Проблемы роста экосистемы

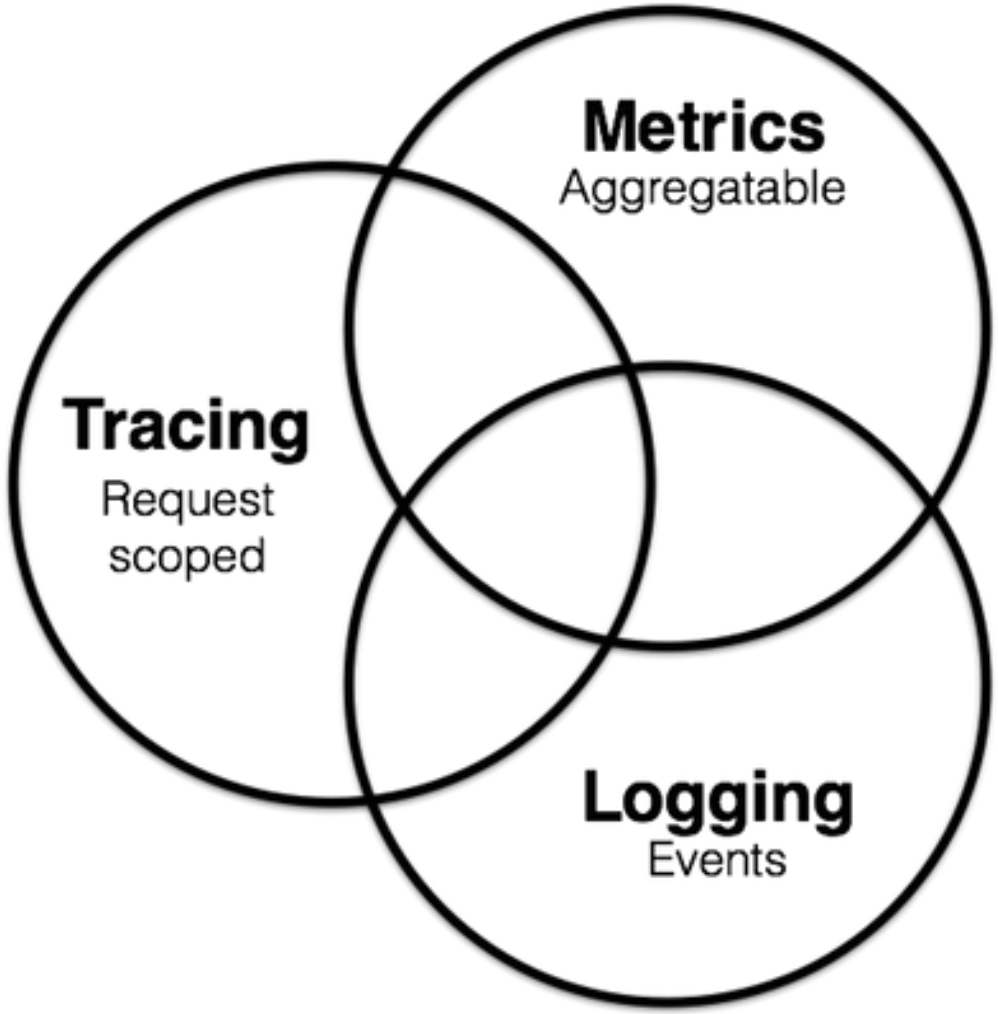
С ростом количества продуктов в экосистеме:

- Усложняется диагностика интеграционных проблем;
- Растет стоимость контроля качества;
- Растет время вывода продукта на рынок.

Как достичь наблюдаемости и снизить сложность?

Фундаментальные типы данных

Единый связный массив данных,
позволяющий проводить анализ



Уровни ландшафта

Количество продаж снизилось,

потому, что **процесс продажи** завершался с ошибкой,

из-за сбоя в **сервисе оплаты,**

так как закончилось **место на диске**

Способы обеспечения наблюдаемости

Инвазивные

выше точность

выше влияние

Неинвазивные

ниже точность

ниже влияние

- Observability as a code (Jaeger)
- APM агент

- Сбор и анализ сетевого трафика
- Аналитика по базе данных
- Машинный анализ логов

Коробочное решение или набор инструментов

Критерии:

- Стоимость
- Кастомизация
- Vendor lock-in
- Готовность развивать экспертизу
- On-premise vs cloud

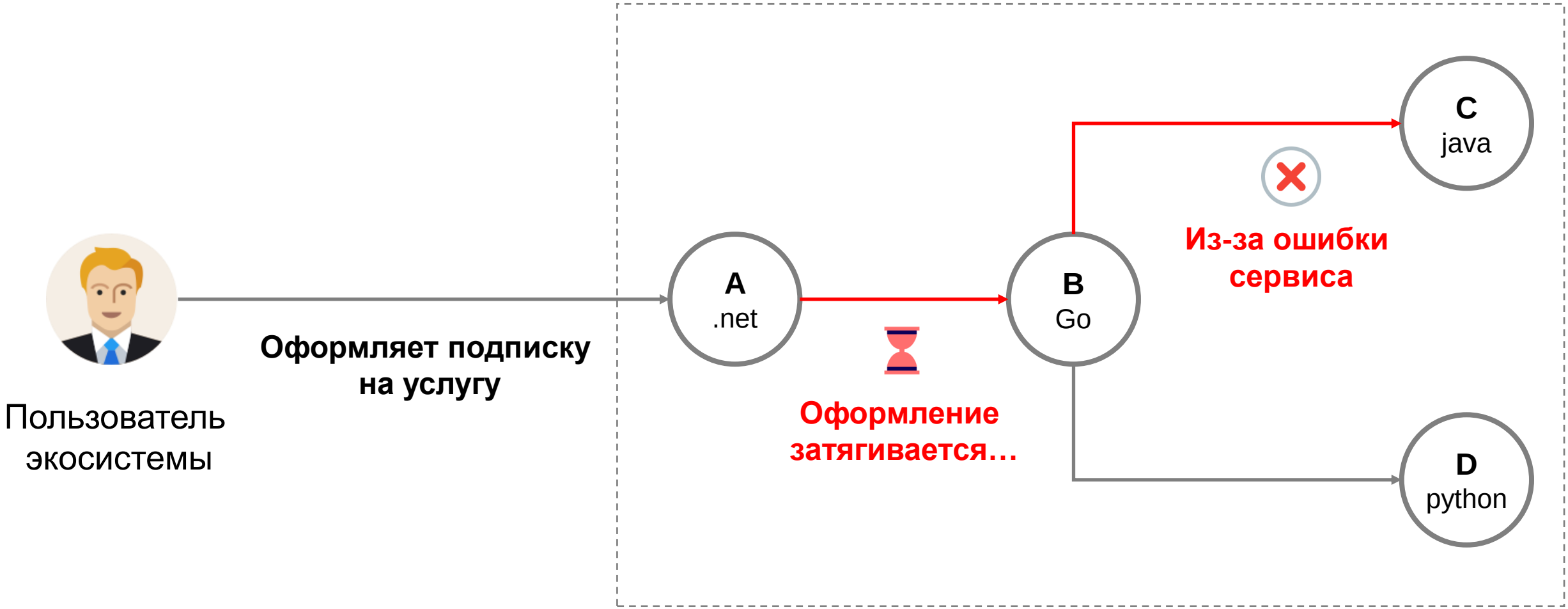
Платформа Наблюдаемости

Набор инструментов для решения задач наблюдаемости в экосистеме МТС

- Цель - наблюдаемость экосистемы
- Platform as a Service
- Единое хранилище данных
- Операционные задачи - диагностика
- Аналитические задачи - инсайты

Как локализовать ошибку?

Быстро понять – кто виноват



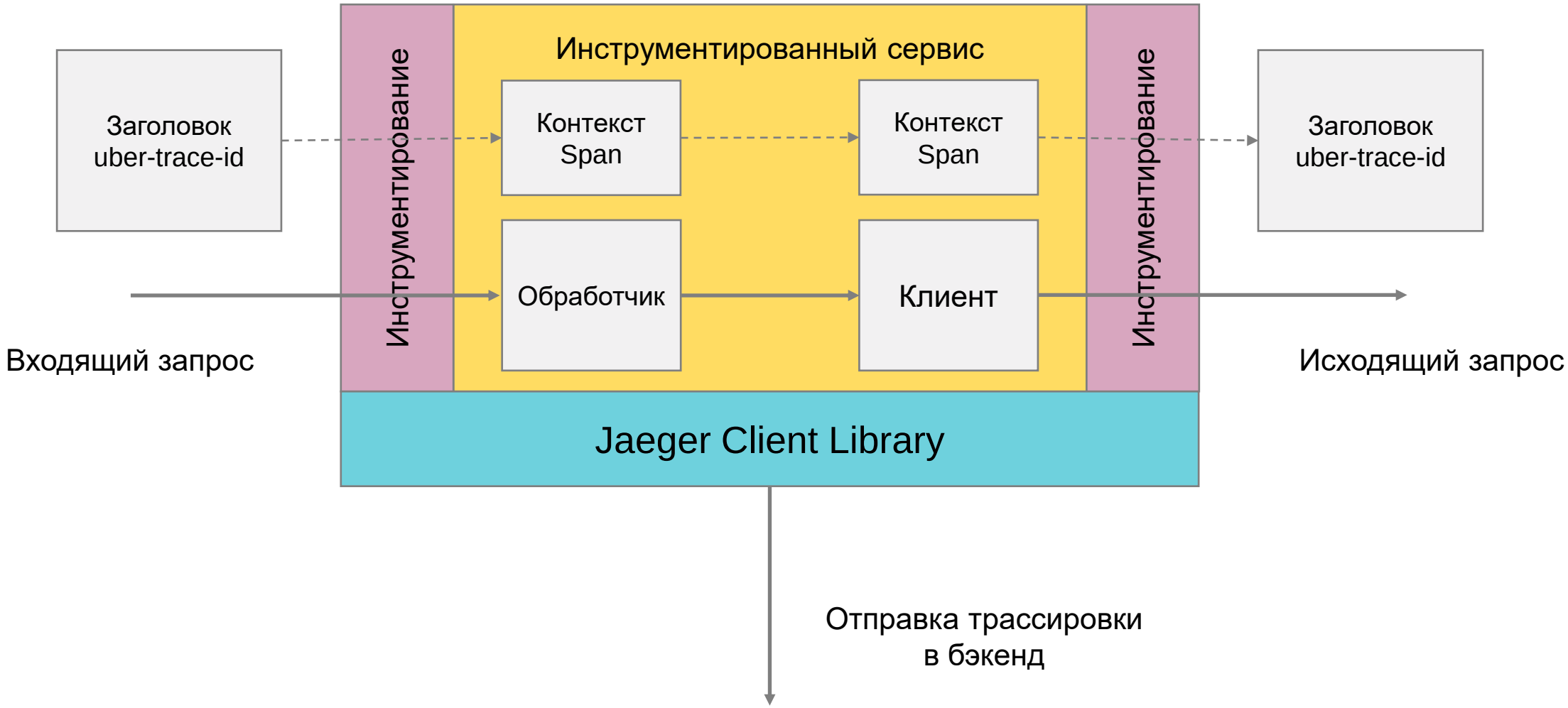
Хотим быстро понять, что во всем виноват продукт C

Распределенная трассировка

Причинно-следственная связь событий (дерево процесса)

Trace:

Схема работы



Системы распределенной трассировки

Jaeger

- Поддержка нескольких бэкендов
- Базируется на OpenTracing
- Совместимость с OpenTelemetry
- Совместимость с Open Zipkin
- Поддержка Remote sampling
- Apache 2.0 license

Elasticsearch APM

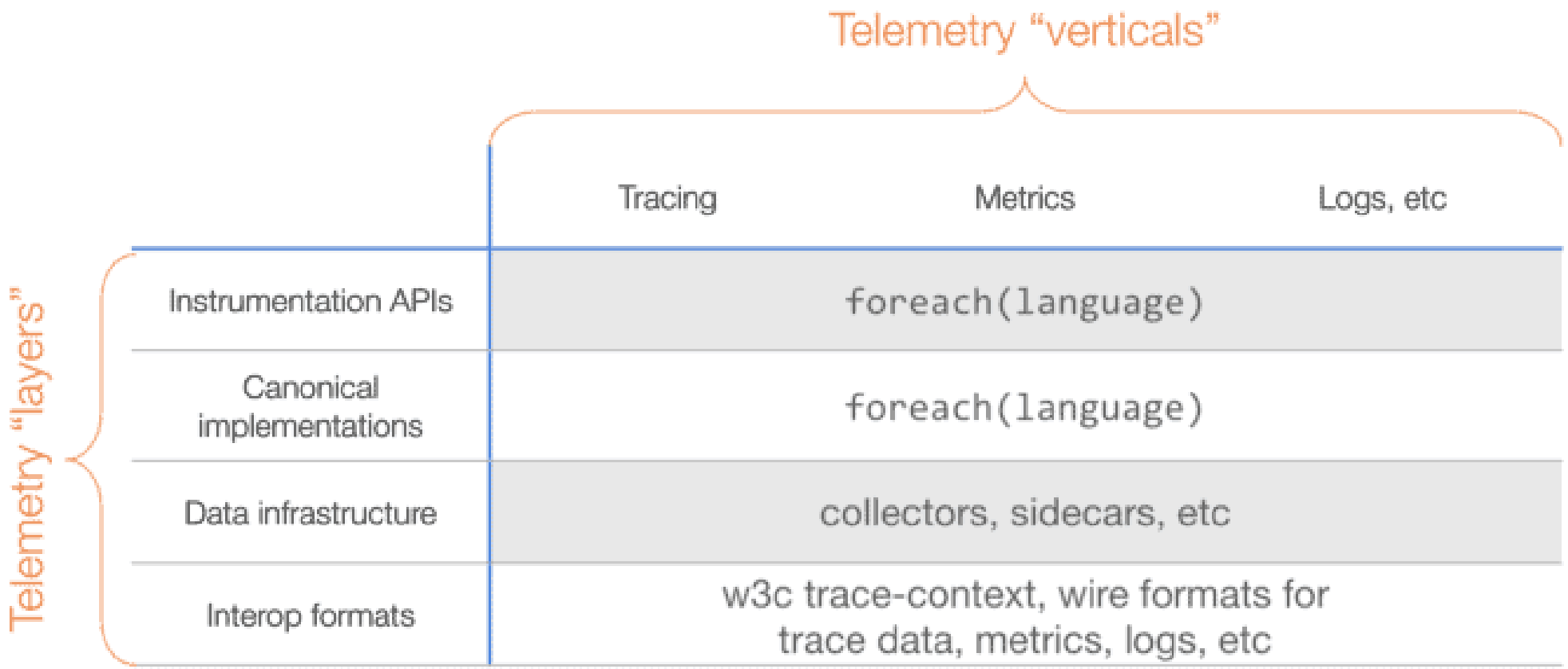
- Трассировка и некоторые метрики
- Совместимость с OpenTracing
- Совместимость с OpenTelemetry
- Корреляция логов
- Поддержка Remote sampling
- Apache 2.0 license для клиентов
- SSPL / Elasticsearch license на сервер

Вывод:

1. Elasticsearch предлагает более комплексное решение
2. Мы выбрали Jaeger из-за простоты модификации и возможности “сменить поставщика”

Как избежать Vendor lock-in

Стандарты и модульность

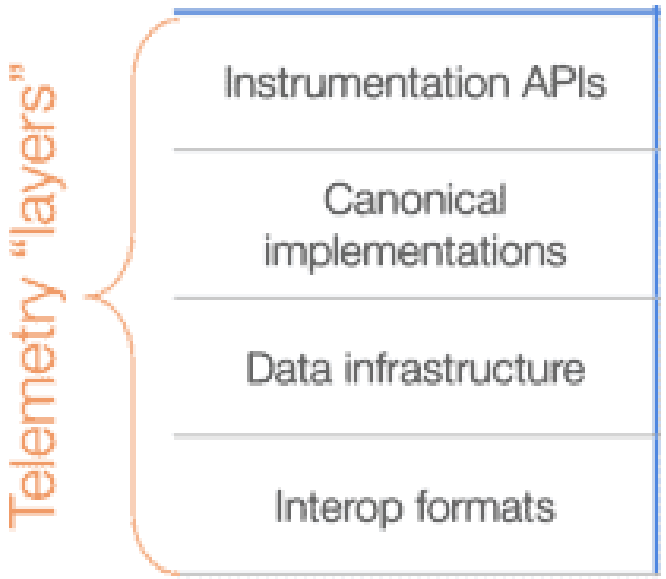


Стандарты логирования

Нет единого стандарта, но есть встроенные механизмы и популярные библиотеки.

В .NET есть **Microsoft.Extensions.Logging**:

- Scope
- Структурное логирование



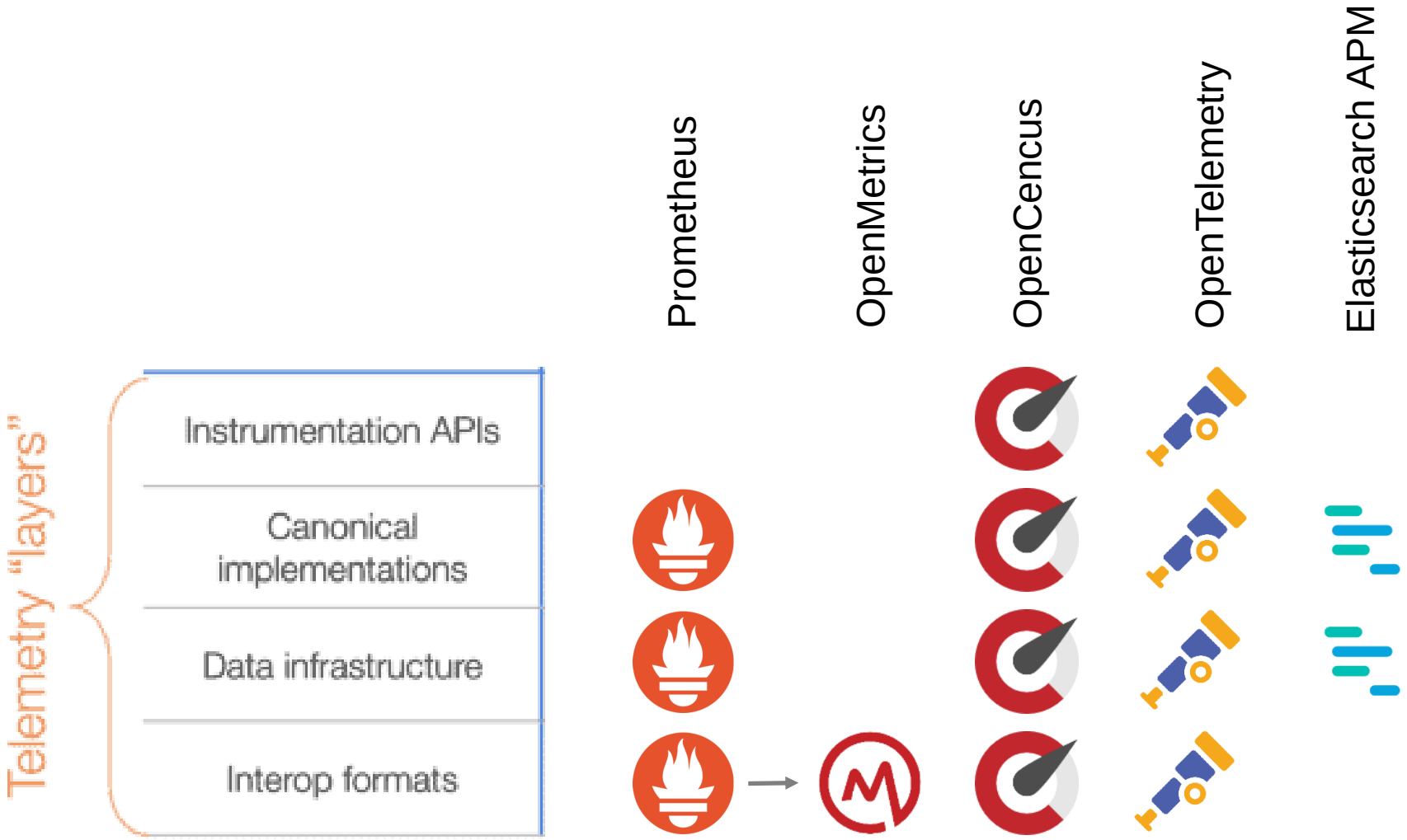
ILogger, ILoggerProvider, ILoggerFactory

Serilog, log4net, NLog, …

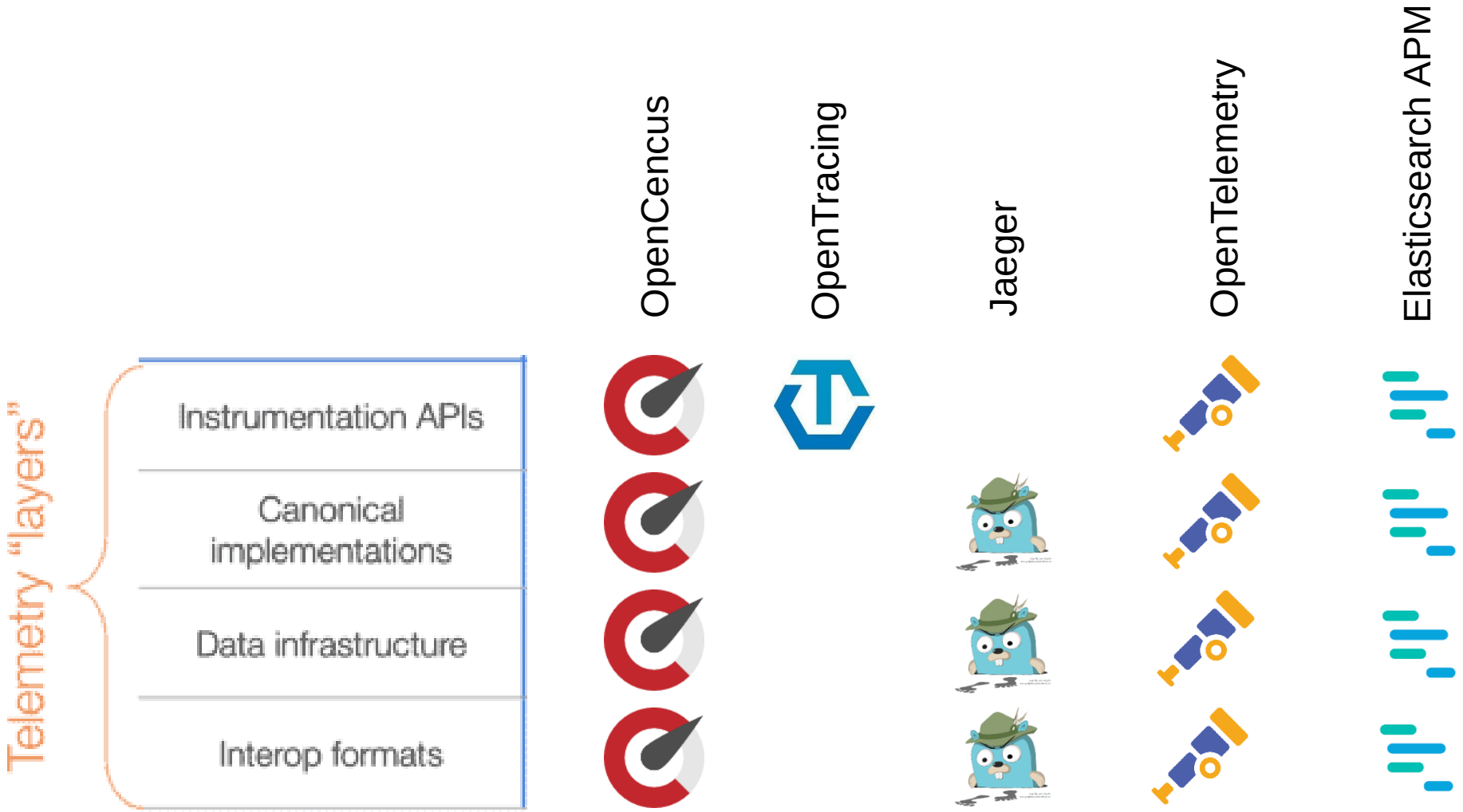
fluentd, filebeat, logstash …

json, plain text …

Стандарты сбора метрик



Стандарты распределенной трассировки



Проброс контекста

OpenZipkin

X-B3-TraceId: 80f198ee56343ba864fe8b2a57d3eff7
X-B3-ParentSpanId: 05e3ac9a4f6e3b90
X-B3-SpanId: e457b5a2e4d86bd1
X-B3-Sampled: 1

Jaeger

uber-trace-id: 6570c6d578b3c800:7500f18ceb180533:62a0a59694e97707:1

traceID

spanID

sampled flag

Elasticsearch APM

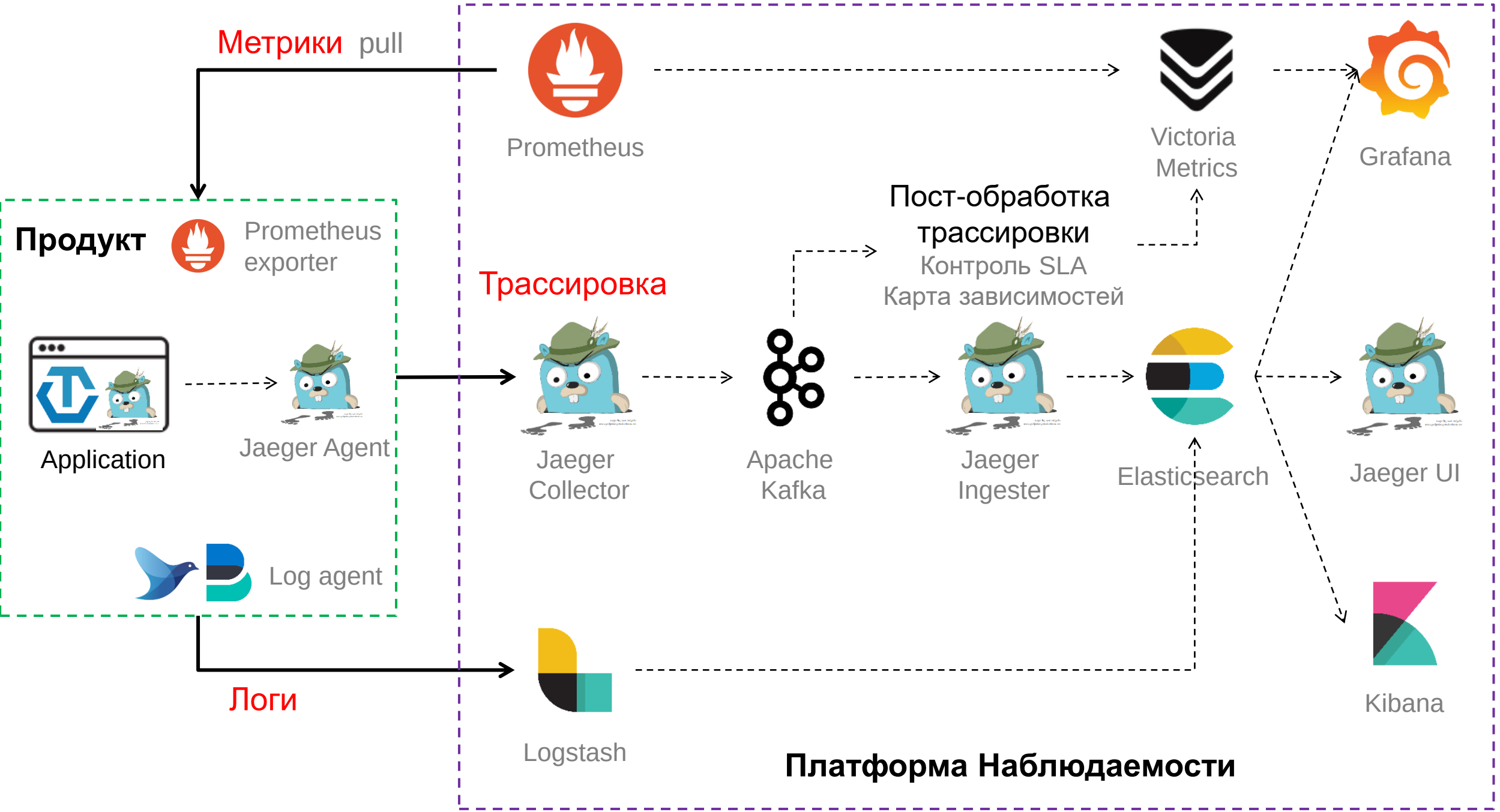
elastic-apm-traceparent: 00-f109f092a7d869fb4615784bacefcfd7-5bf936f4fcde3af0-01

W3C

Recommendation

traceparent: 00-0af7651916cd43dd8448eb211c80319c-00f067aa0ba902b7-01
tracestate: rojo=00f067aa0ba902b7,congo=t61rcWkgMzE

Tracer – specific


Метрики pull
Prometheus
Victoria Metrics
Grafana
Продукт
Prometheus exporter
Пост-обработка трассировки
Контроль SLA
Карта зависимостей
Трассировка
Application
Jaeger Agent
Jaeger Collector
Apache Kafka
Jaeger Ingester
Elasticsearch
Jaeger UI
Log agent
Логи
Logstash
Kibana
Платформа Наблюдаемости

Как унифицировать подход?

OpenTelemetry

OpenTracing + OpenCensus = OpenTelemetry

Трассировка | Трассировка и метрики | Трассировка, метрики, логи

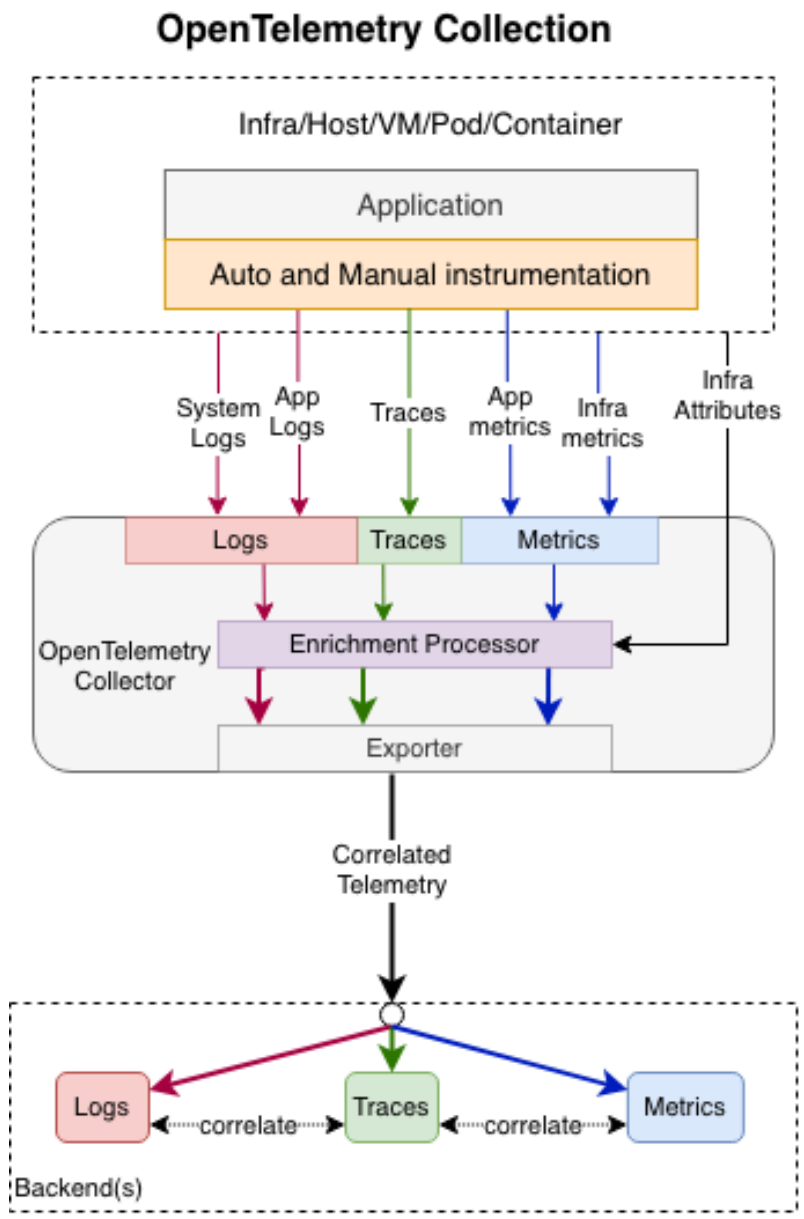
OpenTelemetry

Швейцарский нож в области наблюдаемости.
Набор API для работы с трейсами, метриками и логами.

Преимущества:

- Не зависит от реализации (бэкенда)
- Широко принят вендорами (Elasticsearch APM, NewRelic, AppDynamics)
- Поддержка в .NET
- Совместимость с OpenTracing и OpenCencus
- Одна инфраструктура для всех типов данных

Tracing	Metrics	Logging
Stable	Experemental	Experemental



Инструментирование в .NET

В .NET уже есть механизм трассировки - **System.Diagnostics.Activity**

OpenTelemetry		.NET
Span	=	Activity
Tracer	=	ActivitySource

```
private static readonly ActivitySource MyActivitySource =
  new ActivitySource("MyCompany.MyProduct.MyLibrary");

public static void Main()
{
    using (var activity = MyActivitySource.StartActivity("SayHello"))
    {
        activity?.SetTag("foo", 1);
        activity?.SetTag("bar", "Hello, World!");
        activity?.SetTag("baz", new int[] { 1, 2, 3 });
    }
}
```

New in .NET 5:

- ActivityContext
- ActivityLink
- ActivityEvent
- ActivityKind

Уже инструментированы:

- ASP.NET Core
- EFCore
- HttpClient

Как использовать на продуктиве?

Высоконагруженные системы

Наблюдаемость – не бесплатное удовольствие, особенно для высоконагруженных систем.

Стратегия сэмплирования	Как работает	Поддержка
Constant	Все или ничего Пример: полный сбор данных	Jaeger, OpenZipkin, Elastic APM
Probabilistic	Вероятностный сбор Пример: сбор 10% трейсов	Jaeger, Elastic APM
Rate Limiting	Ограничение потока трейсов Пример: не более 2 трейсов в секунду	Jaeger, OpenZipkin
Remote (Adaptive)	Централизованное управление динамическая настройка Пример: 10% для модуля A и 2 трейса/секунду для B	Jaeger, Elastic APM

Влияние на систему

Показатель	Влияние
RAM	Незначительный рост +100 KB
CPU	На уровне шума, <1%
Время отклика	Без изменений
Network	Незначительное увеличение

Реальный опыт:

- Продуктивный контур
- .net 4.7.2
- ~600 TPS
- Windows
- 100% сбор трассировки

Ошибки отправки Jaeger не влияют на приложение, но возможна потеря данных при сетевых проблемах.

Подробность трассировки

Нужно балансировать между подробностью трейса и удобством анализа.

Трейс – это средство **локализации** интеграционных проблем!

Проблемы больших трейсов (> 1000 спанов**)**:

- Фризы интерфейса при отображении
- Большая вложенность
- Сложный поиск
- Нагрузка на бэкенд

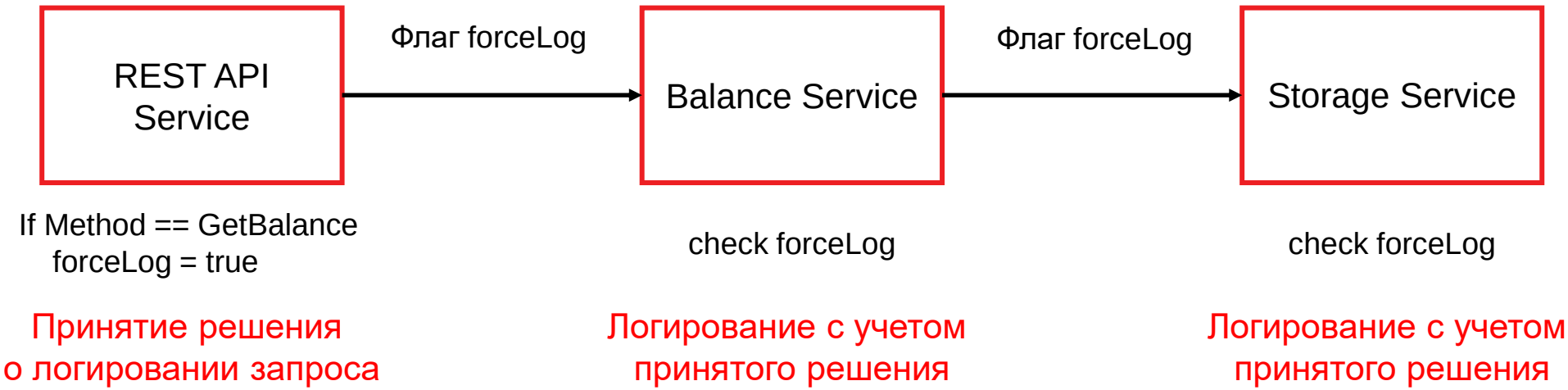
Как найти корневую причину ошибки?

Проблематика логов

- Сбор логов оказывает влияние на производительность (риск аварии)
- Большой объем хранения (дорого хранить)
- Слабая структурированность (долго искать)
- Отсутствие контекста / сложность корреляции событий (долго искать)

Выборочный сбор логов

Нужен компромисс между влиянием на систему и подробностью логов.



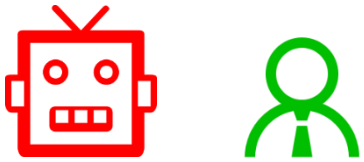
Текущий подход:

- Простые условия сбора
- Привязка фильтра к определенному шагу процесса (однократное вычисление)
- Распространение “решения” фильтра через baggage

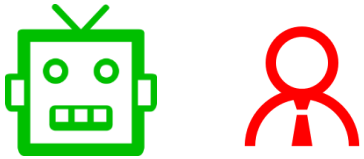
Структурное логирование

Логи должны быть понятны и человеку и машине.

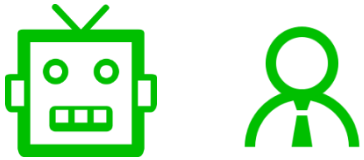
- Снижает требования к системе хранения
- Позволяет группировать события по типу
- Позволяет проводить аналитику и строить визуализации



```
log("User %s logged in from %s", username, ipAddress);
// 2016-05-27T13:02:11.888 User alice logged in from 123.45.67.89
```



```
log({eventId: "user_logged_in", username: username, ip_address: ipAddress });
// { "time": "2016-05-27T13:02:11.888",
// "eventId": "user_logged_in",
// "username": "alice",
// "ip_address": "123.45.67.89“ }
```

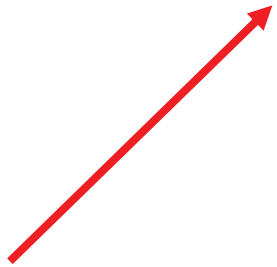


```
log("User {username} logged in from {ip_address}", username, ipAddress)
// { "time": "2016-05-27T13:02:11.888",
// "template": "User {username} logged in from {ip_address}",
// "username": "alice",
// "ip_address": "123.45.67.89" }
```

Привязка логов к контексту

Обогащение лога контекстом трассировки позволяет переходить от локализации к диагностике – от трассировки, к логам.

Time	Message	SpanID	TraceID	LogLevel
August 28th 2019, 14:10:41.525	2019-08-28 14:10:41,525[64] ERROR ArchiveSendingProcessor - ArchiveQueueId 3220: end with error.	eb798b4a79a3ce	10846ed40516dd5	ERROR



Логи привязаны к контексту
(Request scoped logs)

Полнотекстовый поиск

Не требует усилий от команды, но потребляет больше ресурсов.

Elasticsearch

Полнотекстовый поиск,
индексация ключей и сообщения

Grafana Loki

Индексация ключей
(меток)

- Высокое потребление ресурсов
- Быстрый поиск по ключам
- Быстрый поиск по телу сообщения

- Низкое потребление ресурсов
- Быстрый поиск по меткам
- Медленный поиск по телу сообщения

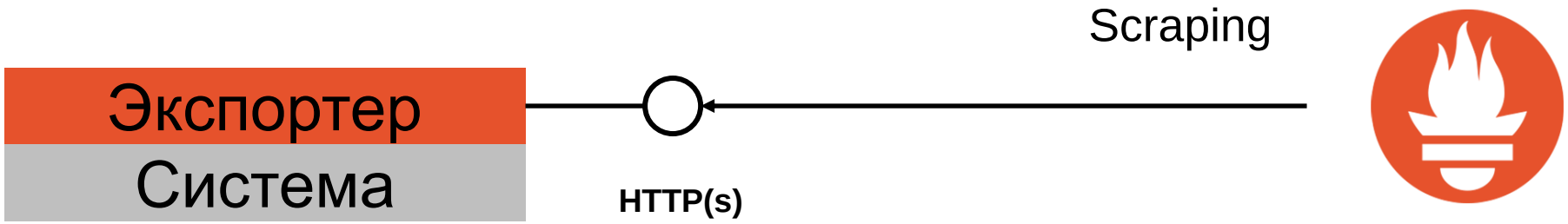
Продуманное структурное логирование
может снизить стоимость хранения логов

Экспортеры Prometheus

Что хотим собирать:

- Метрики железа (CPU, RAM, IO)
- Метрики операционных систем (windows / unix)
- Метрики сторонних компонент (базы данных, очереди, веб-сервера ..)
- Метрики приложения

Экспортер – приложение, способное собирать метрики из источника и выставлять их в виде конечной точки в формате Prometheus.



Экспортеры Prometheus для .net

Пакет **prometheus-net** – API для выставления собственных метрик

Типы метрик:

- **Counter** – возрастающая последовательность
- **Summary** – кол-во запросов, обработанных за последние 10 минут
- **Gauge** – кол-во запросов в очереди
- **Histogram** – распределение запросов по длительности

Готовые пакеты:

- prometheus-net.AspNetCore
- prometheus-net.AspNetCore.HealthChecks
- prometheus-net.AspNetCore.Grpc
- prometheus-net.NetFramework.AspNet

Как контролировать качество?

Модель качества продукта

Качество системы
ISO/IEC 25010:2011
ГОСТ Р ИСО/МЭК 25010-2015

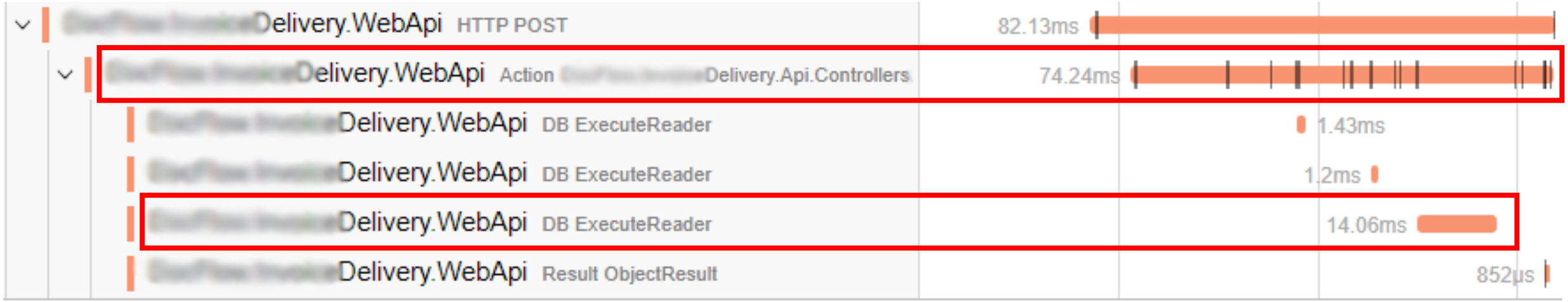
Функциональная пригодность	Уровень производительности	Удобство использования	...	Защищенность

- Функциональная корректность
- Полнота
- Целесообразность

- Временны́е характеристики
- Использование ресурсов
- Уровень производительности

Пример для ASP.NET Core и EFCore

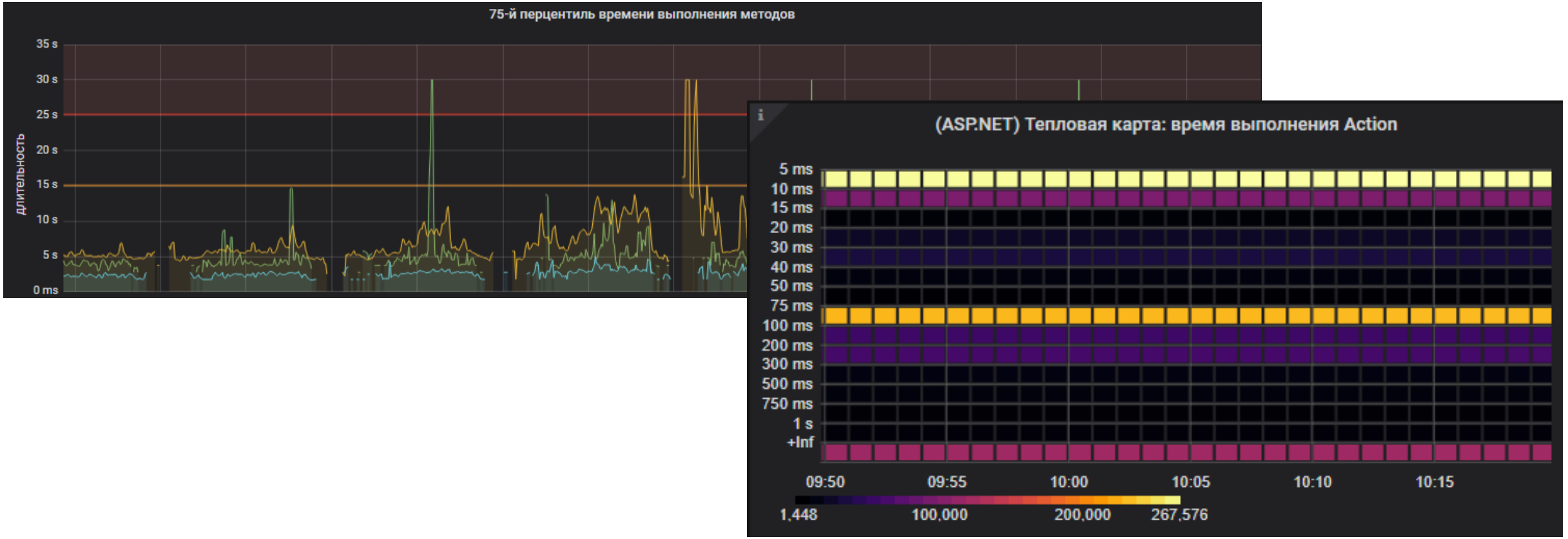
Длительность и ошибки каждого Action фиксируется в трассировке.



Дашборд “временные характеристики”

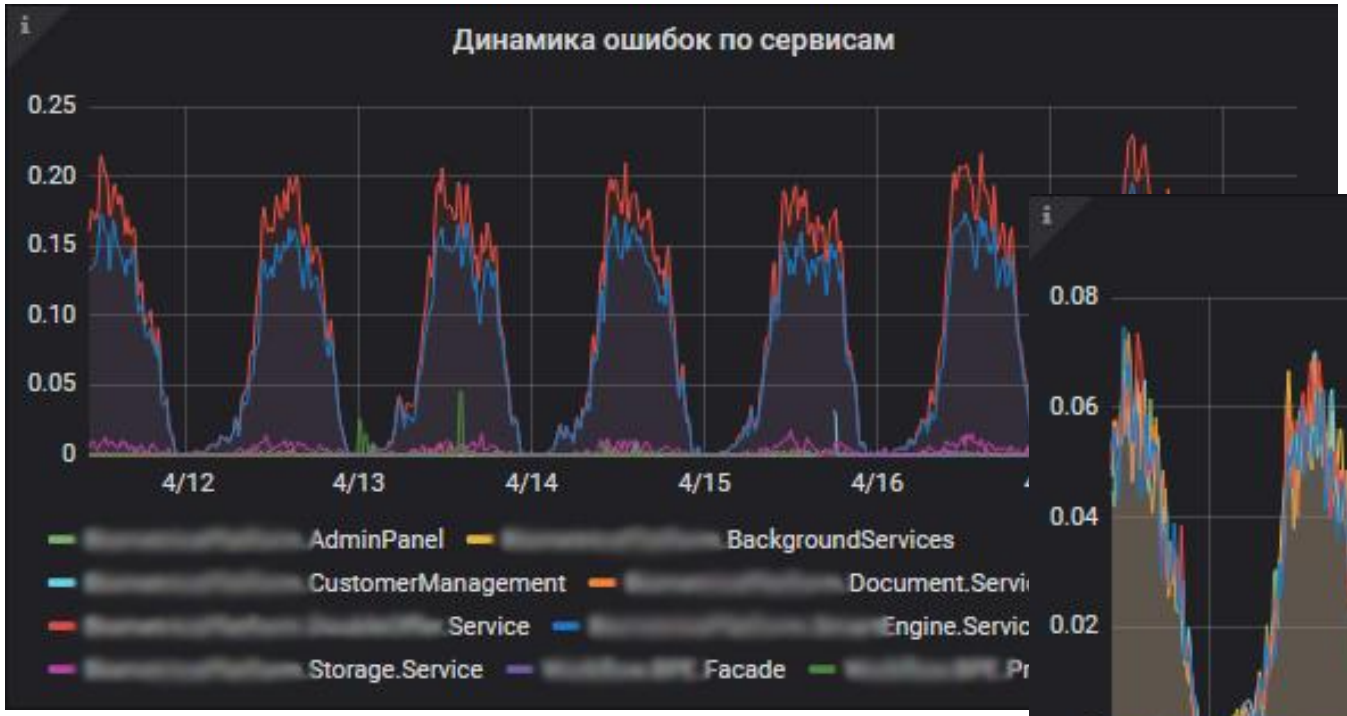
Тепловая карта и перцентиль длительности выполнения

- ASP.NET Core action
- Запросов Entity Framework Core



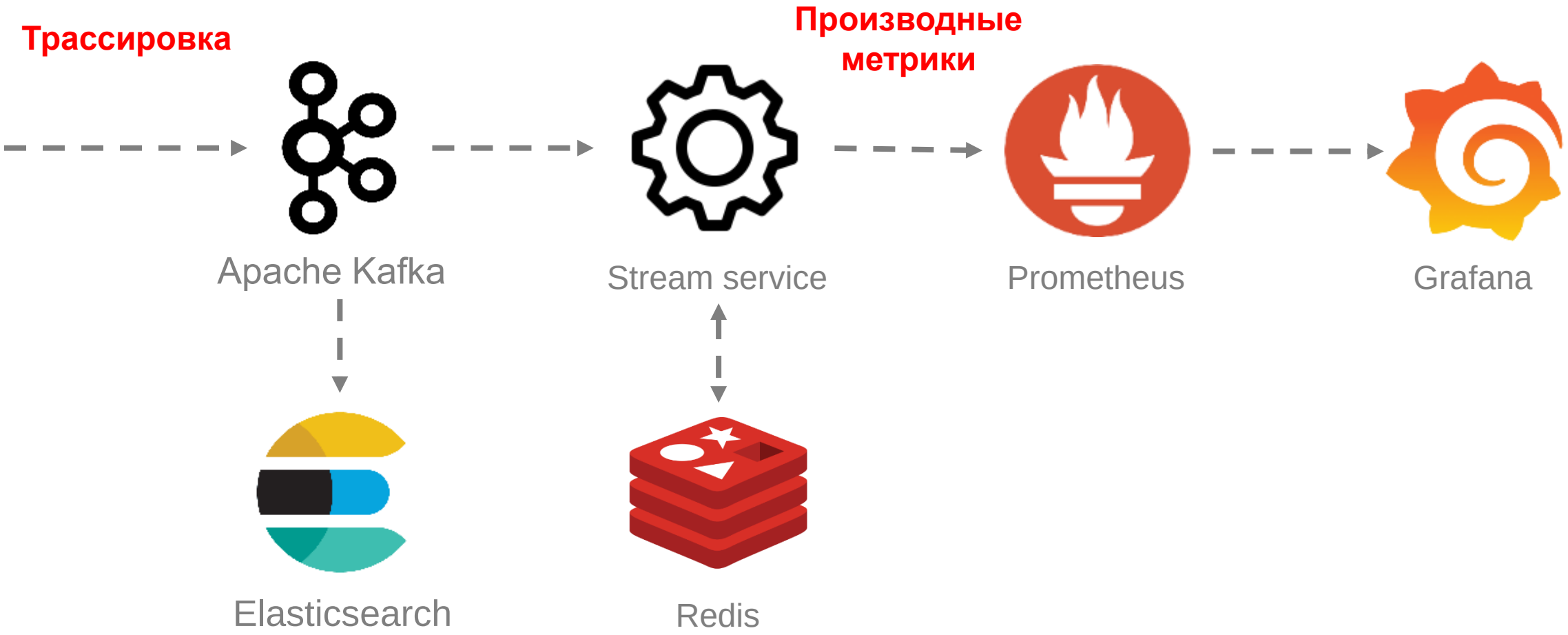
Дашборд “функциональная корректность”

У каждого спана есть теги с признаком ошибки, именем хоста и сервиса.
Отсюда дашборд с количеством ошибок по сервисам и хостам.



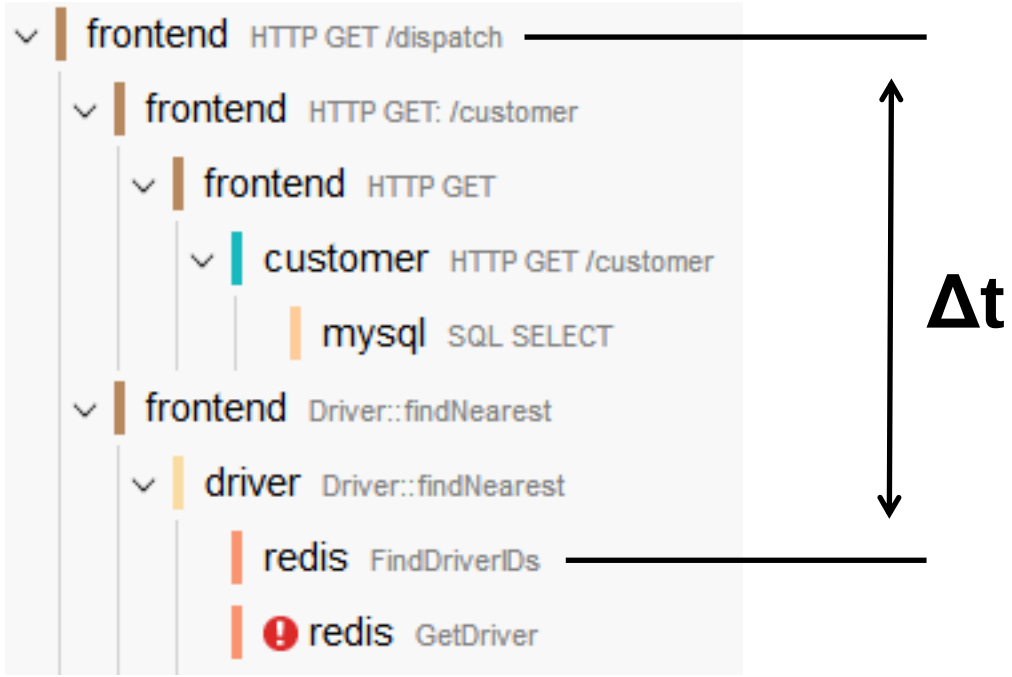
Пост обработка спанов

Постобработка позволяет считать производные метрики из трассировки



Контроль SLA процессов

Измеряем время от нажатия кнопки “Купить”, до отправки чека на почту.



Контроль времени между произвольными точками:

- Требует промежуточного хранилища для сессии
- Порядок событий не гарантирован
- Требует решения проблемы синхронизации времени на хостах

Как визуализировать архитектуру системы?

Карта IT ландшафта

Анализ связей между спанами в распределенной трассировке.

Сущности:

- ИТ-продукты
- Сервисы
- Сервера

Как предотвращать аварии?

4 золотых сигнала мониторинга

Задержка

Время обработки запроса

99й перцентиль длительности спана обработки запроса в **распределенной трассировке**

Насыщение

Достижение предела производительности

Достижение порогового значения **метрик** потребления CPU / RAM / …

Трафик

Нагрузка на систему

Количество спанов обработки запроса в **распределенной трассировке**

Ошибки

Ошибки обработки запросов

Количество записей **логов** с уровнем ERROR или FATAL

White box vs black box monitoring

White box

Real user monitoring

Метрики обработки запросов реальных пользователей

Black box

Синтетический мониторинг

Робот, опрашивающий систему, имитирующий пользователя

- Способен выявлять проблему заранее
- Дает больше информации для локализации

- Создает меньше шума и реагирует на важные симптомы

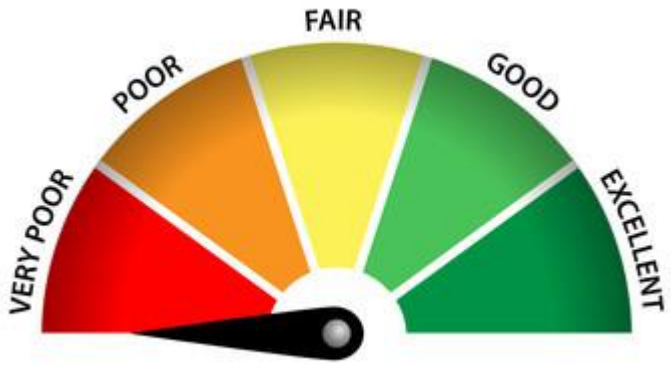
Нужны оба типа мониторинга

Выводы

Стоимость обеспечения наблюдаемости

Дорого и сложно

- Потенциальные потери из-за низкого качества продукта
- Часы разработчиков, тестировщиков и инженеров для ручной диагностики и тестирования



Сложно и дорого

- Лицензии
- Железо для обработки и хранения
- Часы разработчиков и инженеров для настройки

Баланс наблюдаемости

Уровень наблюдаемости должен расти вместе со зрелостью приложения.

Критерии:

- **Требования к качеству** (SLA, доступность, производительность)
- **Сложность** (кол-во сервисов, модулей, типов транзакций, команд)

Список литературы

Список литературы

1. **Gartner Magic Quadrant 2021 APM** - https://www.appdynamics.com/resources/reports/gartner-magic-quadrant-apm
2. **W3C Trace Context Specification -** https://www.w3.org/TR/trace-context/
3. **OpenTelemetry Tracing API for .net -** https://github.com/open-telemetry/opentelemetry-dotnet/blob/main/src/OpenTelemetry.Api/README.md#introduction-to-opentelemetry-net-tracing-api
4. **ГОСТ Р ИСО/МЭК 25010-2015 -** https://docs.cntd.ru/document/1200121069
5. **Google SRE Book -** https://sre.google/sre-book/monitoring-distributed-systems/#xref_monitoring_golden-signals
6. **Mastering Distributed Tracing -** https://www.amazon.com/Mastering-Distributed-Tracing-performance-microservices-ebook/dp/B07MBNGF7Q