Введение в distributed tracing

Панычев Дмитрий, CIO OBI

При участии Безкоровайный Денис, Proto, директор подразделения DevOps/DevSecOps

Зачем вам использовать distributed tracing?

Distributed Tracing – это способ понять, как проходят запросы в распределенной системе

Дает возможность *контроля* и *понимания* распределенной системы:

- Через какие сервисы проходят транзакции?
- Сколько времени обрабатываются запросы в том или ином сервисе?
- Где именно возникают проблемы?

Distributed Tracing и KPI производительности

На основе анализа трейсов можно получить *метрики и KPI* производительности приложений:

- Какой процент вызовов завершились ошибкой?
- Какая длительность исполнения транзакций по 90-му перцентилю?
- Сколько вызовов в секунду обрабатывает сервис X?

Distributed Tracing – плюсы использования

для разработчиков и SRE:

- Понимание всей картины распределенной системы
- Детальный анализ прохождения запросов
- Выявление узких мест
- Возможность просмотреть транзакцию целиком

Зачем. Сетевая связанность с SSO.

Проблема:

- **Не всегда работает SSO;**

Зачем. Сетевая связанность с SSO.

Проблема:

- **Не всегда работает SSO;**
- **Падаем по таймауту;**

Зачем. Сетевая связанность с SSO.

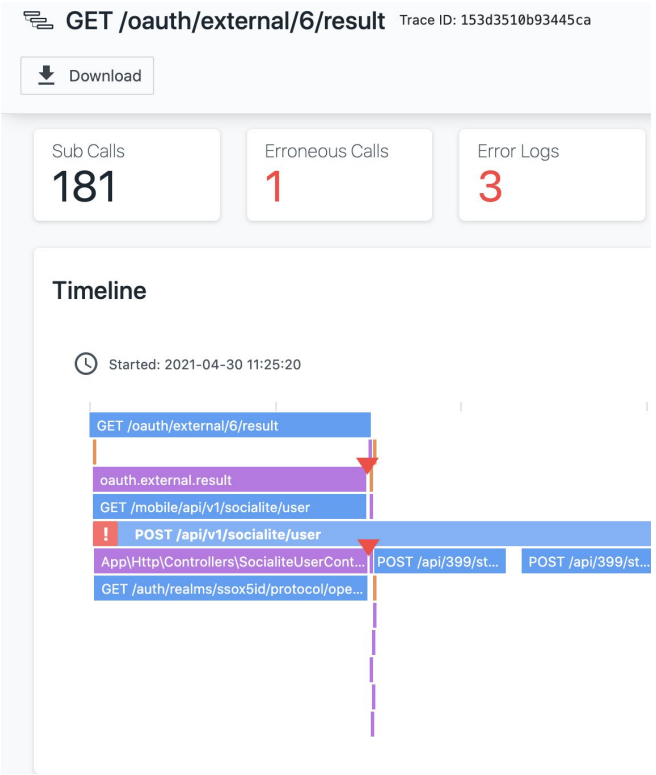
Проблема:

- **Не всегда работает SSO;**
- **Падаем по таймауту;**
- **Но не везде.**

Зачем. Сетевая связанность с SSO.

Исследование:

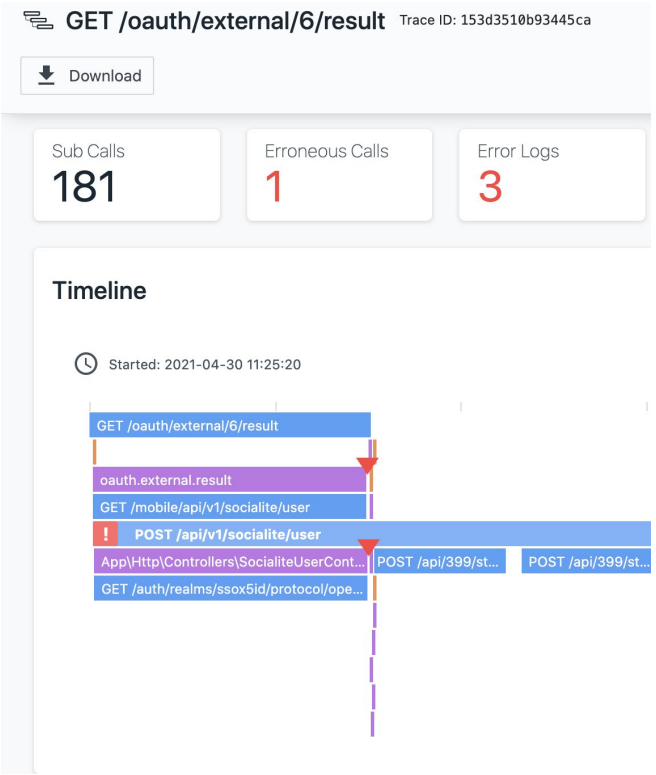
- **Трейс – запрос падает по таймауту;**



Зачем. Сетевая связанность с SSO.

Исследование:

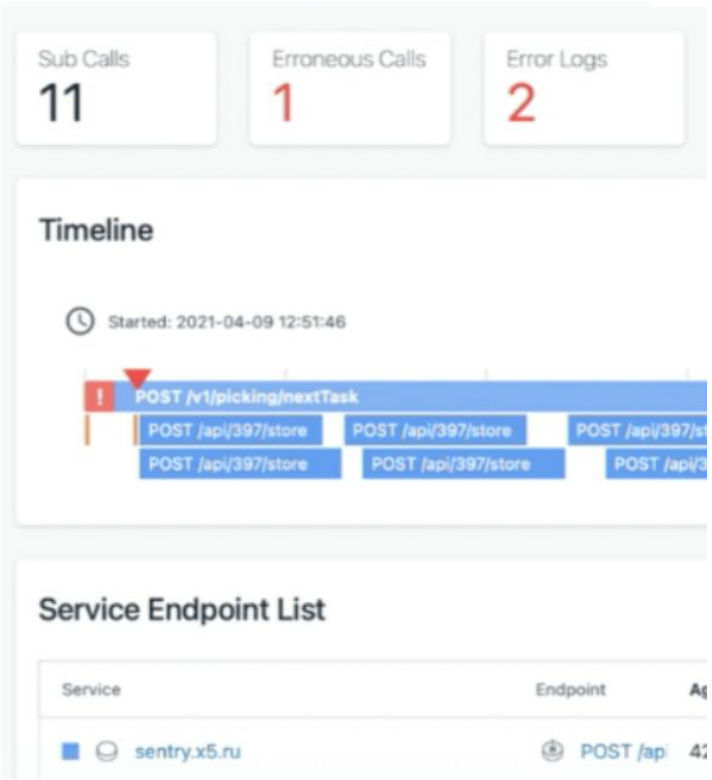
- **Трейс – запрос падает по таймауту;**
- **Обращения к сервису на карте – нет ошибок;**



Зачем. Сетевая связанность с SSO.

Исследование:

- **Трейс – запрос падает по таймауту;**
- **Обращения к сервису на карте – нет ошибок;**
- **В трейсе – таймаут коннекта к Sentry (тот же контур);**

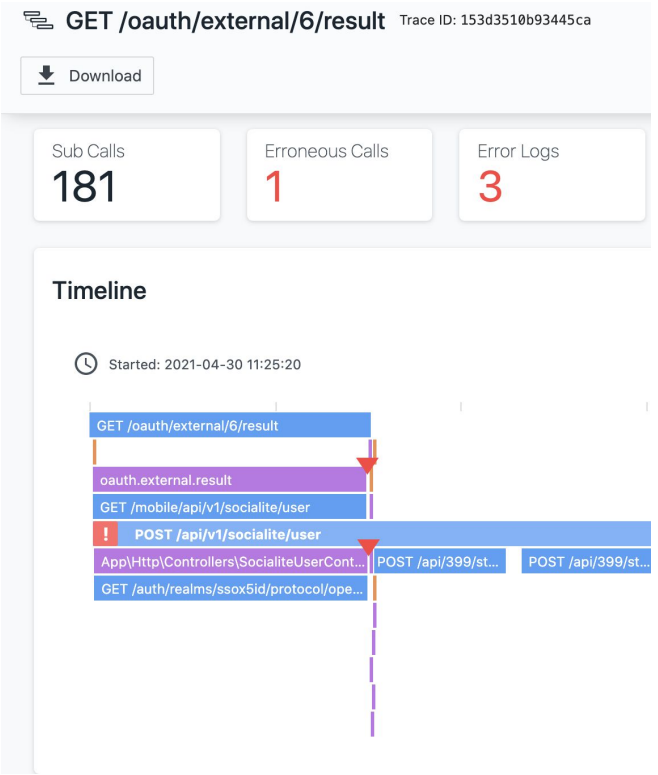


Зачем. Сетевая связанность с SSO.

Исследование:

- **Трейс – запрос падает по таймауту;**
- **Обращения к сервису на карте – нет ошибок;**
- **В трейсе – таймаут коннекта к Sentry;**
- **Вывод – потеря сетевой связанности с внешним контуром.**

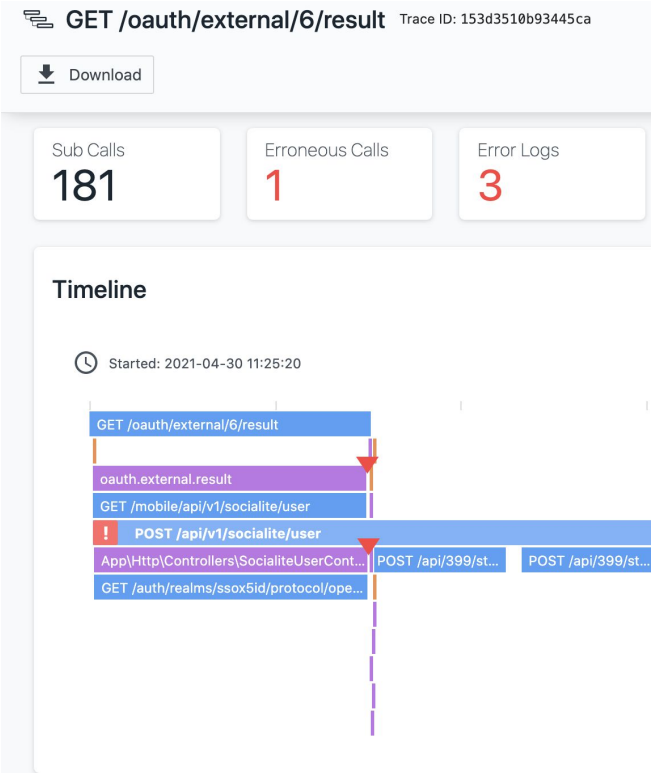
- **Время на исследование проблемы – ~1м**



Зачем. Анализ производительности систем.

...но об этом чуть позже.

А пока – немного теории.



Distributed tracing – часть observability

Observability – это возможность быстро понять систему, ее состояние и возможные причины деградации производительности

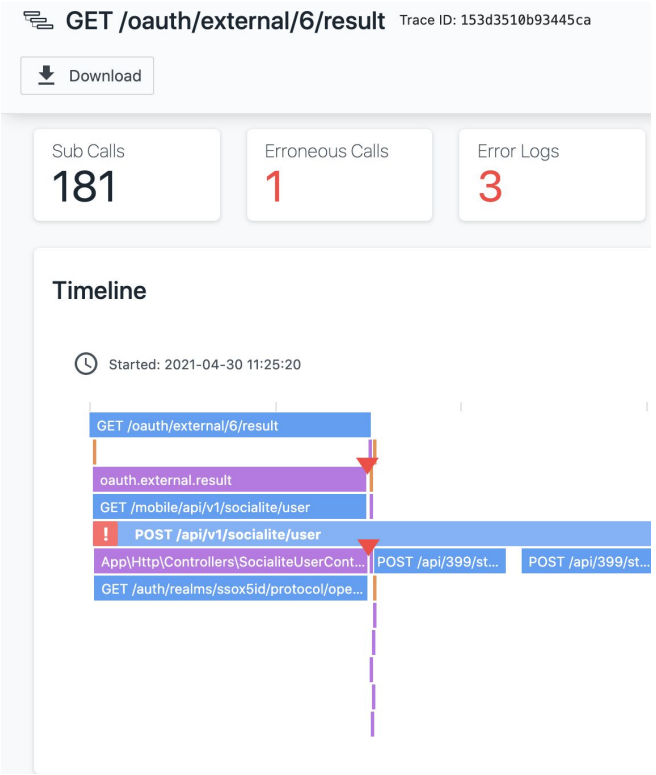
Observability = метрики + трейсы + логи

Из анализа трейсов легко получить метрики

В трейсах уже содержатся логи приложений и сообщения об ошибках

Где проблема-то?..

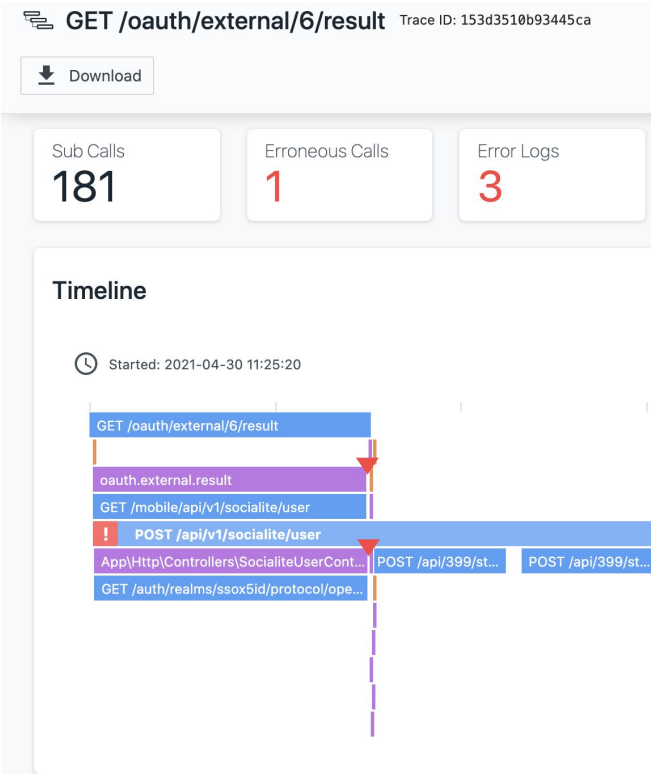
Помните кейс про SSO и сетевую связанность?



Где проблема-то?..

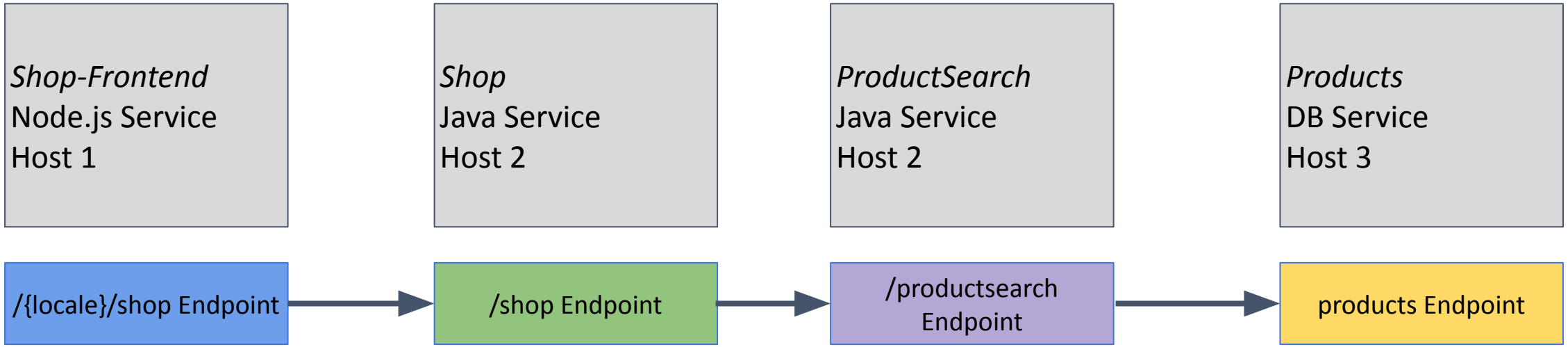
Помните кейс про SSO и сетевую связанность?

Вот там вот!

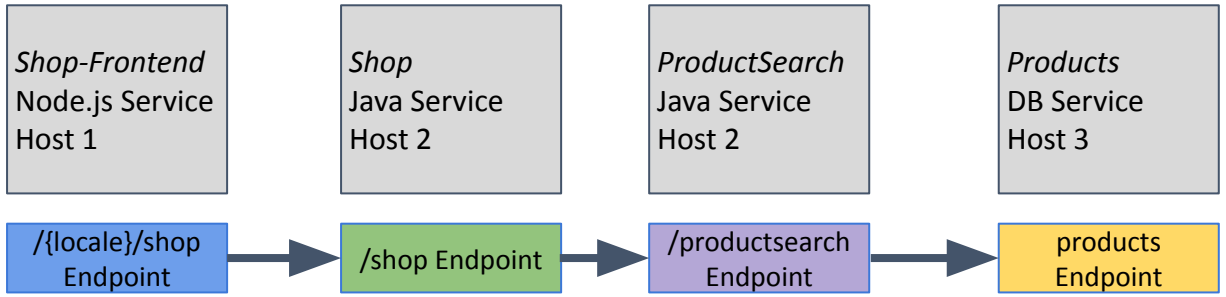


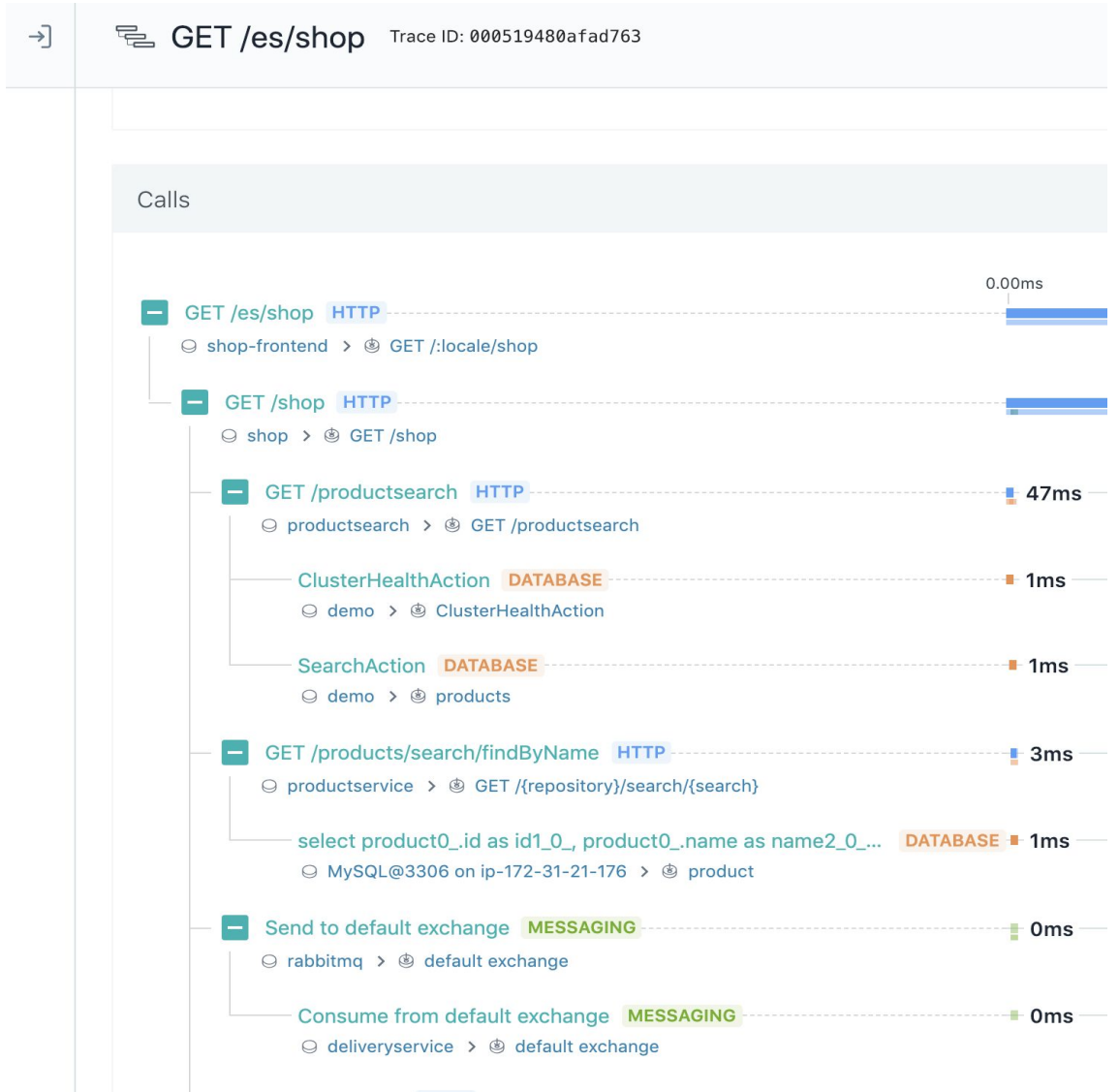
Как работает distributed tracing?

Distributed Tracing: Теория



Distributed Tracing: Теория





Distributed Tracing: Теория

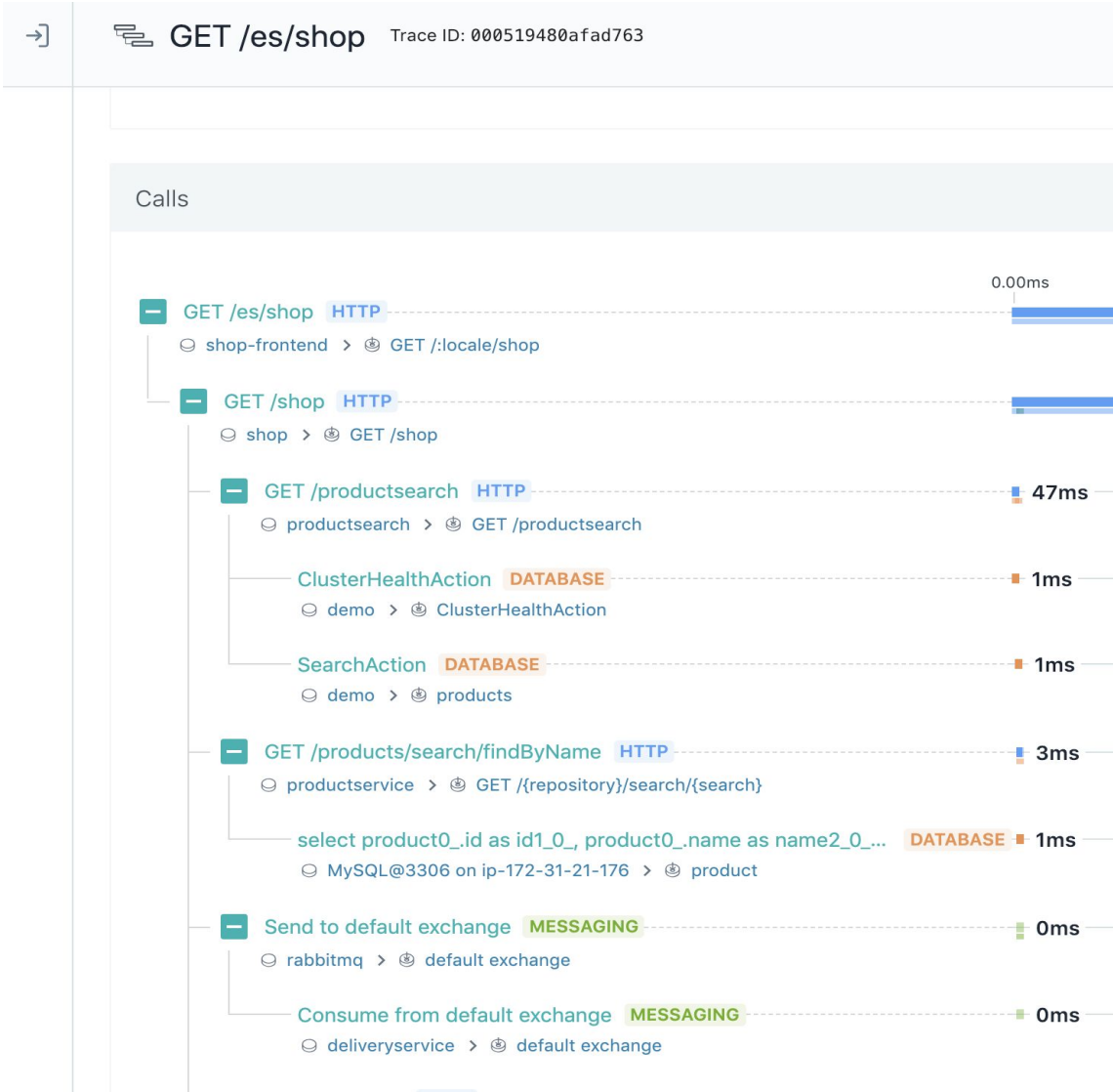
Вызов описывает деятельность в рамках отслеживаемого процесса, как правило запрос между двумя сервисами.

Каждый вызов стоит из одного или нескольких **спанов**.

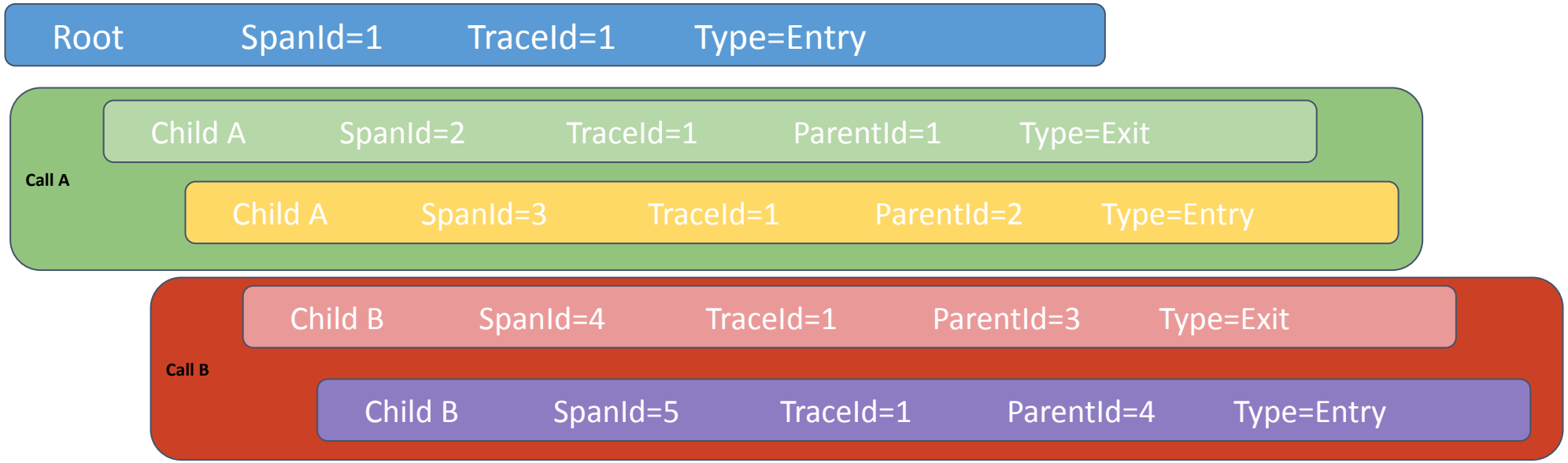
Трейс состоит из одного или нескольких вызовов.

Концепция Трейсов и спанов пришла от:

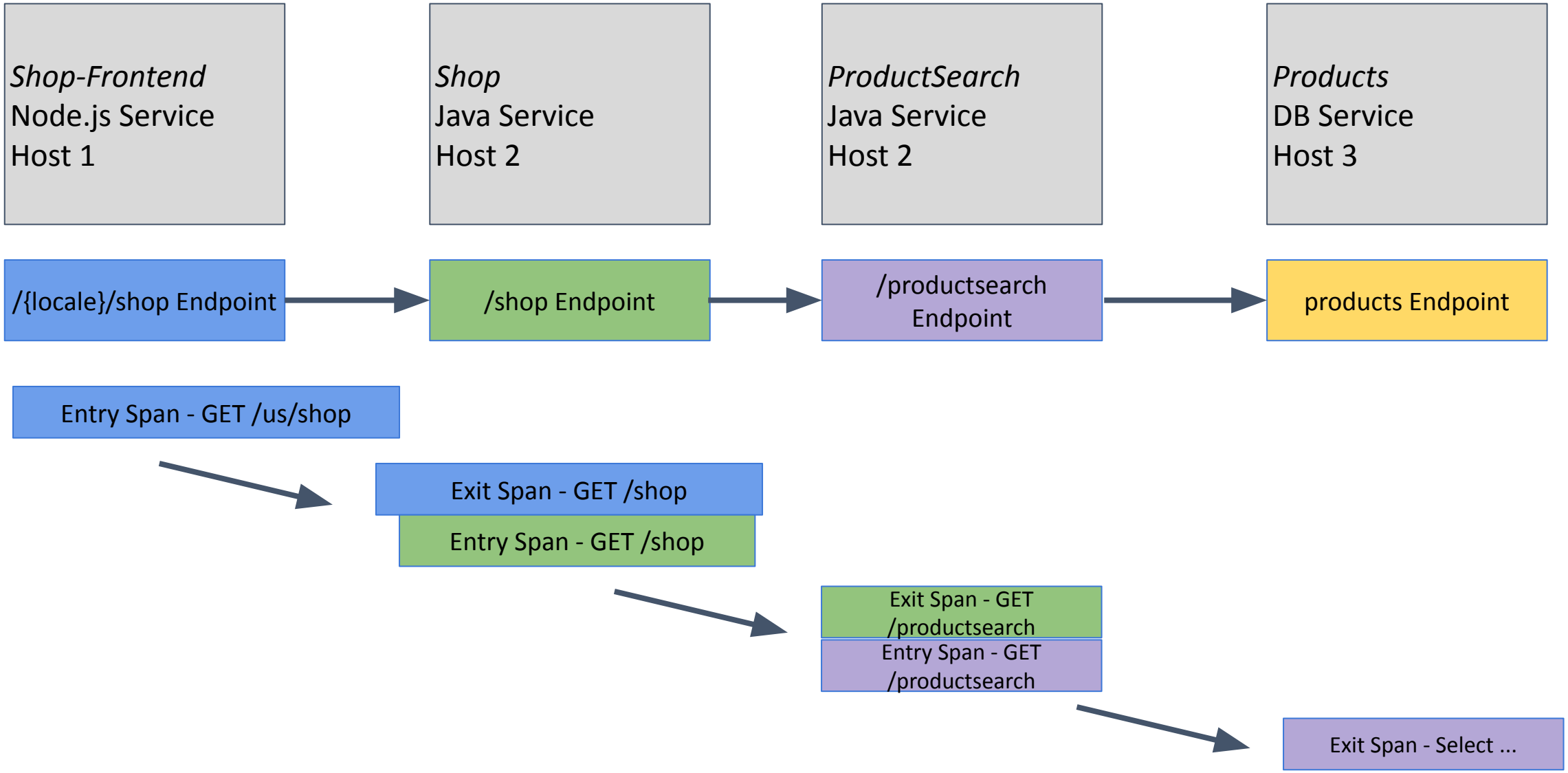
- Google «Dapper, Large-Scale Distributed Systems Tracing Infrastructure» (https://ai.google/research/pubs/pub36356)
- The OpenTracing Semantic Specification: (https://github.com/opentracing/specification/blob/master/specification.md)



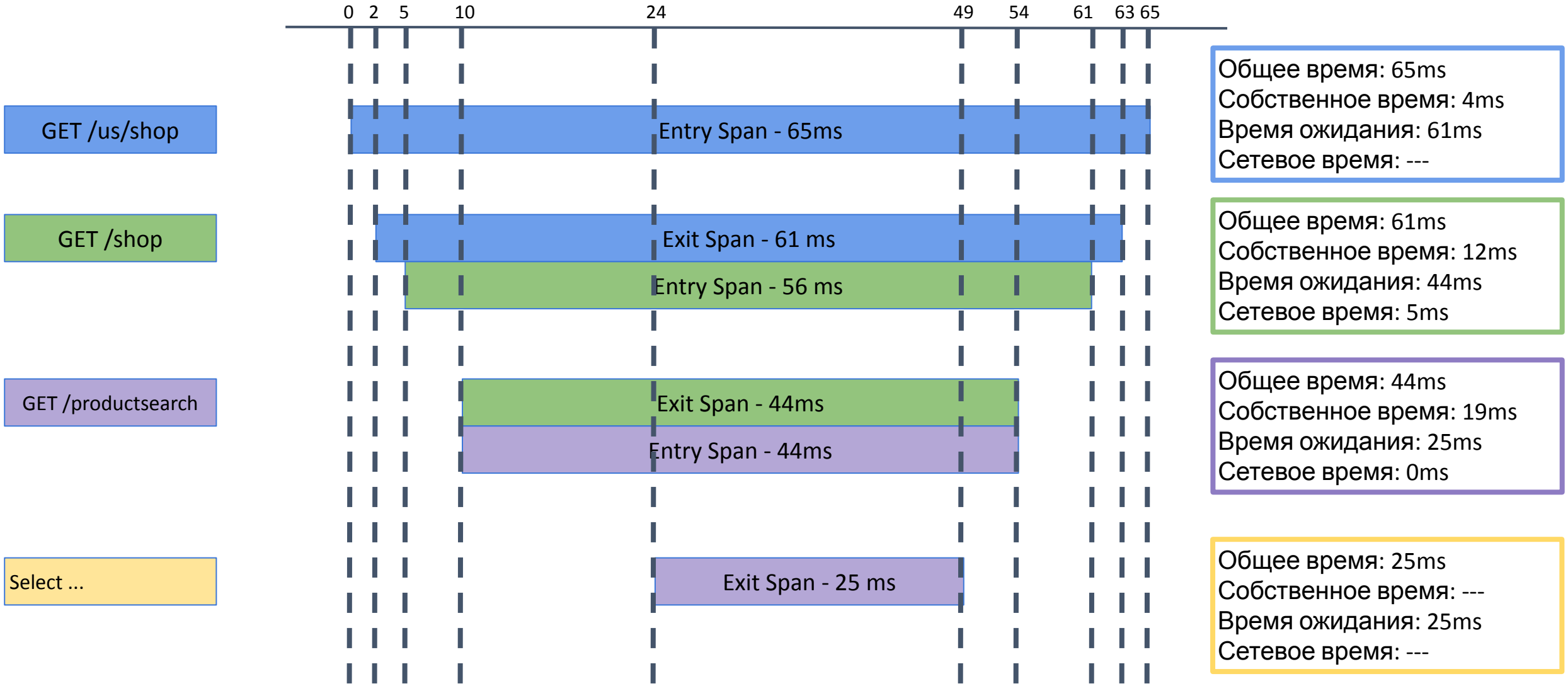
Distributed Tracing: сущности



Пример


Shop-Frontend
Node.js Service
Host 1
Shop
Java Service
Host 2
ProductSearch
Java Service
Host 2
Products
DB Service
Host 3
/{locale}/shop Endpoint
/shop Endpoint
/productsearch Endpoint
products Endpoint
Entry Span - GET /us/shop
Exit Span - GET /shop
Entry Span - GET /shop
Exit Span - GET /productsearch
Entry Span - GET /productsearch
Exit Span - Select ...

Зачем это все нужно? Тайминги!



Общее время: 65ms
Собственное время: 4ms
Время ожидания: 61ms
Сетевое время: ---

Общее время: 61ms
Собственное время: 12ms
Время ожидания: 44ms
Сетевое время: 5ms

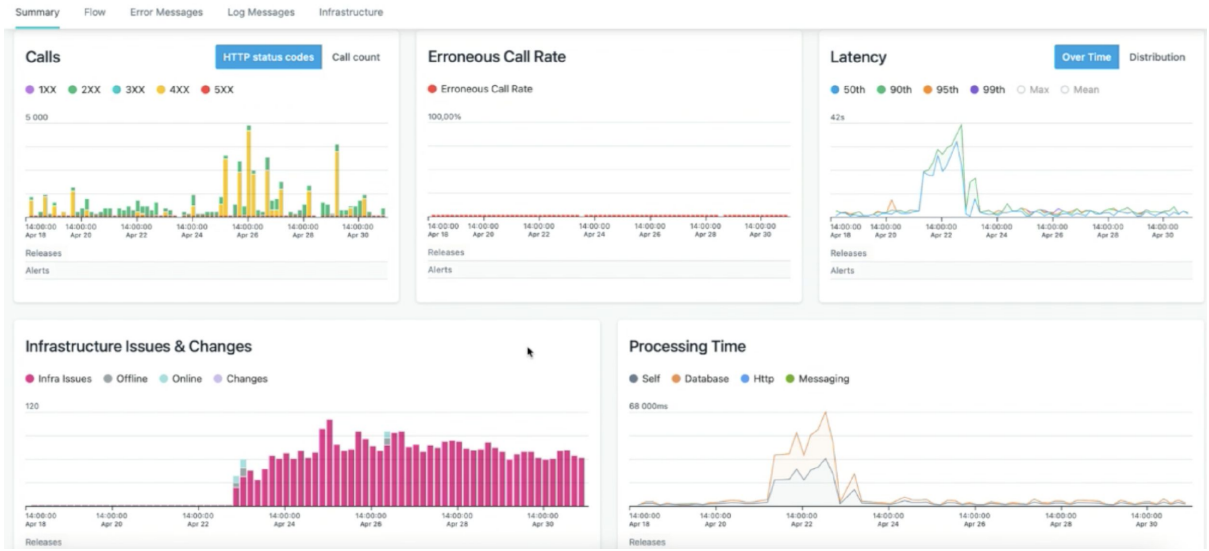
Общее время: 44ms
Собственное время: 19ms
Время ожидания: 25ms
Сетевое время: 0ms

Общее время: 25ms
Собственное время: ---
Время ожидания: 25ms
Сетевое время: ---

И что мы с этим делаем?

Проблема:

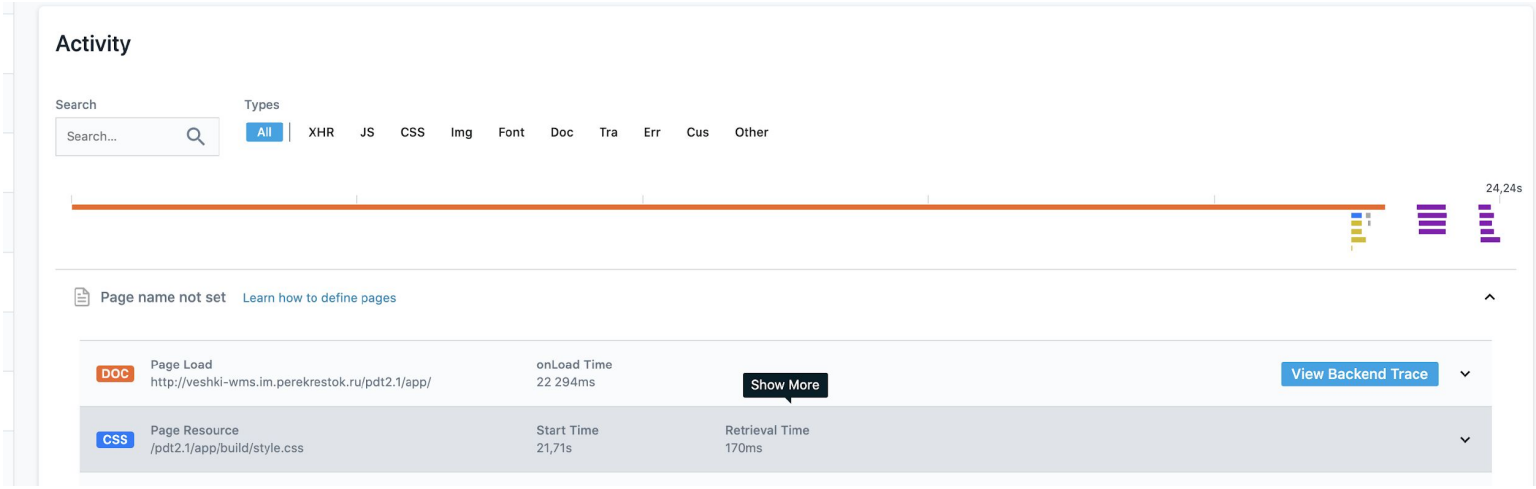
- **Долгое время отбора товаров на WMS (warehouse management system).**



И что мы с этим делаем?

Исследование:

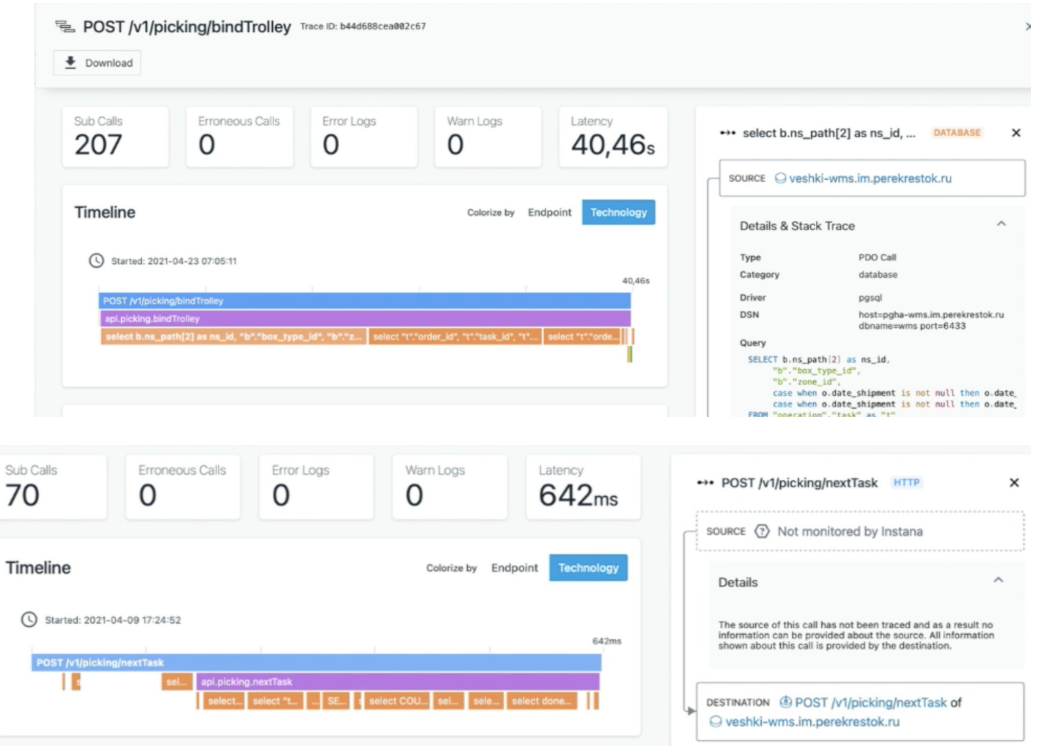
- **Смотрим на трейс;**



И что мы с этим делаем?

Исследование:

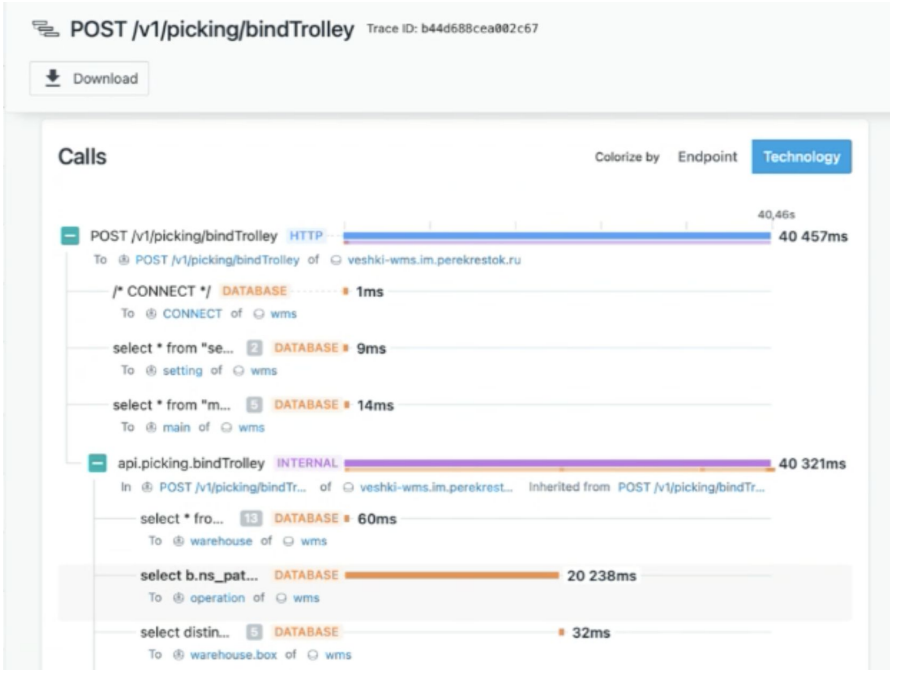
- **Смотрим на трейс;**



И что мы с этим делаем?

Исследование:

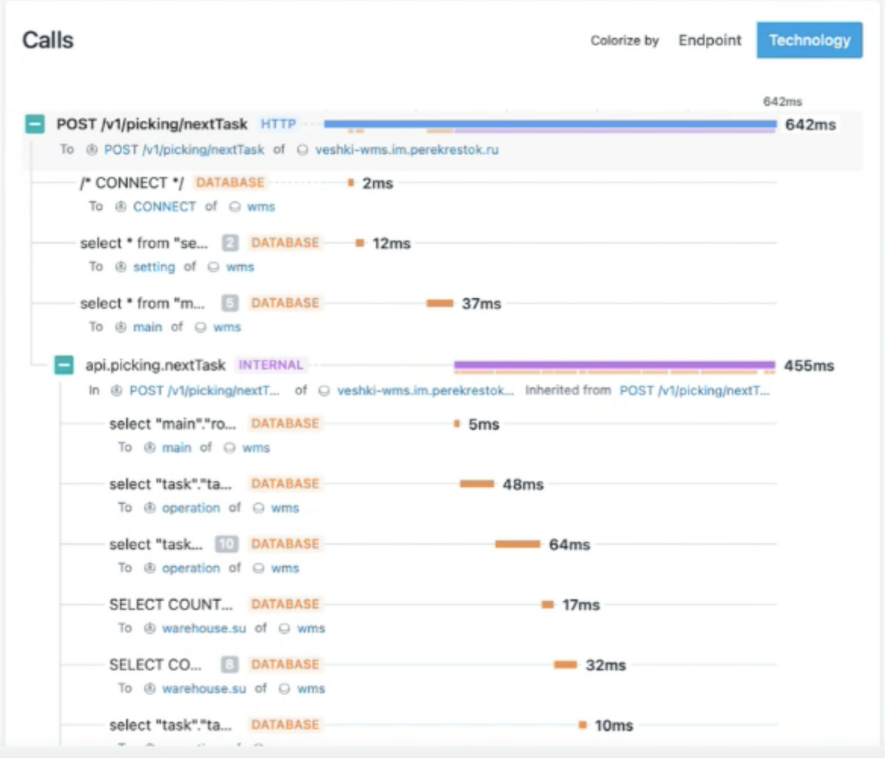
- **Смотрим на трейс;**
- **Находим виновных.**



И что мы с этим делаем?

Исследование:

- **Смотрим на трейс;**
- **Находим виновных.**



Купить? Сделать самому? А может быть – OpenSource?

Варианты?

- Сделай сам (OpenTelemetry+Jaeger/Zipkin);
- Коммерческие APM (POP, AppDynamics, NewRelic, Instana, Dynatrace, etc);
- Бесплатные APM (ELK).

Проблемы и сложности при внедрении

С чем можно столкнуться (обзор?)

- Сложности – объемы данных, сэмплинг трейсов (для снижения нагрузки), детализация трейсов, контекст, разные реализации

- Риски – сроки, ресурсы, влияние на стабильность приложения

С чем столкнулись

Список граблей:

- **Нагрузка на CPU на проде;**
- **Автообновление агента на проде;**
- **Не останьтесь без мониторнга;**
- **Нужно больше места!**

Нагрузка на CPU на проде

Что случилось?

- **На app-нодах (PHP) росло потребление CPU со временем, пропорционально нагрузке;**
- **При отключении агента – возврат в норму.**

Нагрузка на CPU на проде

Кто виноват и что сделали?

- **Виновной признана архитектура агента;**
- **Перезапуск агентов раз в несколько часов;**
- **Конфигурирование агента (отключения ряда функций);**
- **Вендор изменил архитектуру агента.**

Автообновление агента на проде

Что случилось?

- **Фон 500 на сервисе в k8s;**
- **В APM ошибок нет.**

Автообновление агента на проде

Кто виноват и что сделали?

- **Агент автоматически обновился в кластере k8s;**
- **Новая версия агента приводила в некоторых случаях к крашу приложения;**
- **Была зафиксирована версия агента.**

Остаться без мониторинга?

APM – это круто.

Но альтернативные системы нужны.

Нужно больше места!

Трейсы, спаны, метрики.

Представьте, что каждый запрос, проходящий через ваши системы и всё, что он делает – записывается.

Подведение итогов

К чему это всё

- Distributed tracing эффективен для:

К чему это всё

- Distributed tracing эффективен для:
  - Ответа на вечный вопрос «кто виноват»;

К чему это всё

- Distributed tracing эффективен для:
  - Ответа на вечный вопрос «кто виноват»;
  - Понимания работы системы во всех её взаимосвязях;

К чему это всё

- Distributed tracing эффективен для:
  - Ответа на вечный вопрос «кто виноват»;
  - Понимания работы системы во всех её взаимосвязях;
  - Поиска узких мест.

Спасибо!

Панычев Дмитрий, OBI, CIO

- Telegram: @pandmitr
- Facebook: https://www.facebook.com/dmitry.panychev

При участии
Денис Безкоровайный, Proto,
Директор подразделения DevOps/DevSecOps

- Telegram: @d4n1s8
- Facebook: https://www.facebook.com/bezkod