Distributed Tracing для поиска проблем в Entity Framework Core

Стрелов Егор

Ведущий инженер-программист

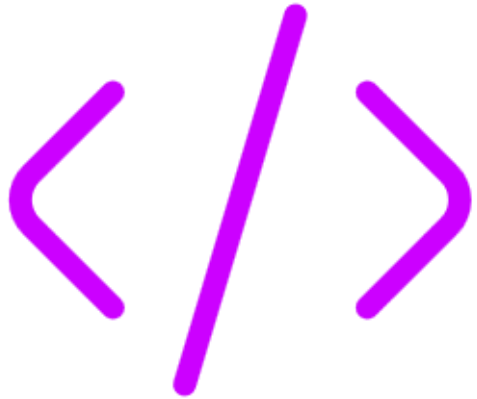
Контур

План доклада

1. Исходная проблема
2. Поиск решения
3. Изменения в приложении
4. Сбор телеметрии
5. Итоги и ограничения

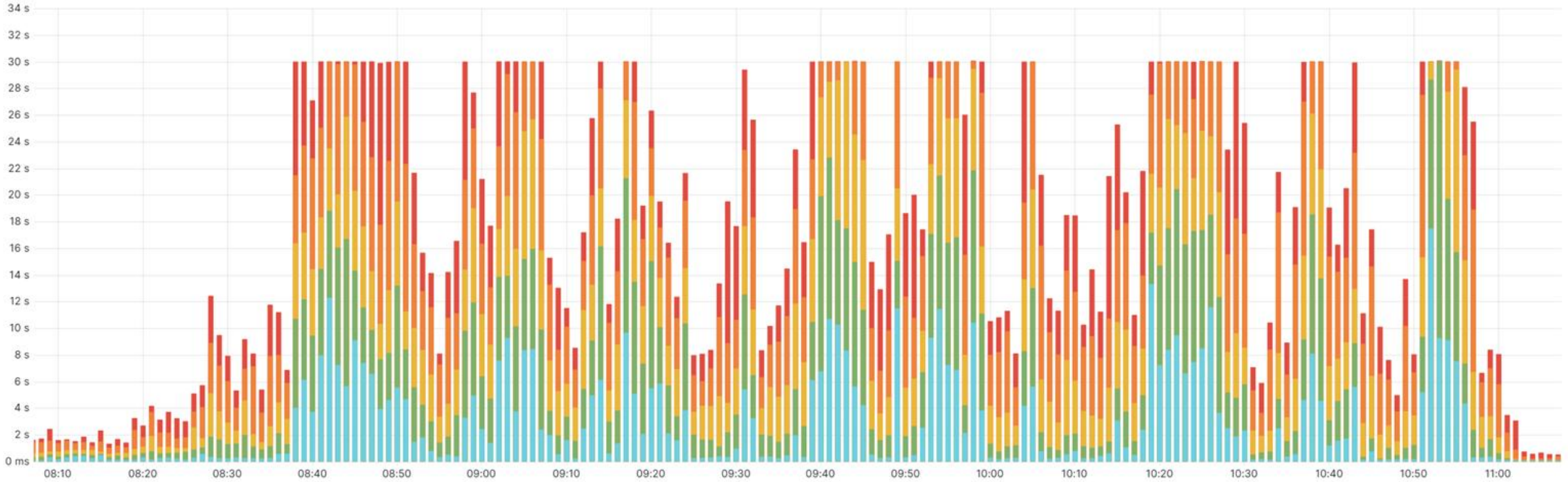
О проекте

- Бэкенд и фоновые сервисы
- Сложная схема базы данных
- Много join операций в запросах
- Рост количества пользователей

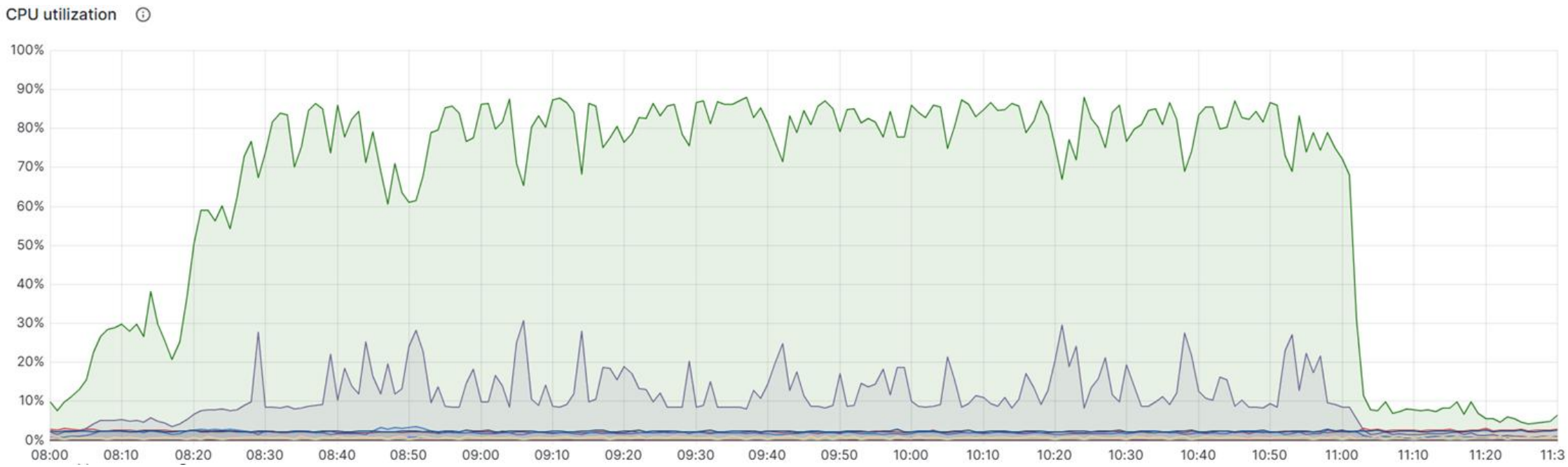


Большое время ответа

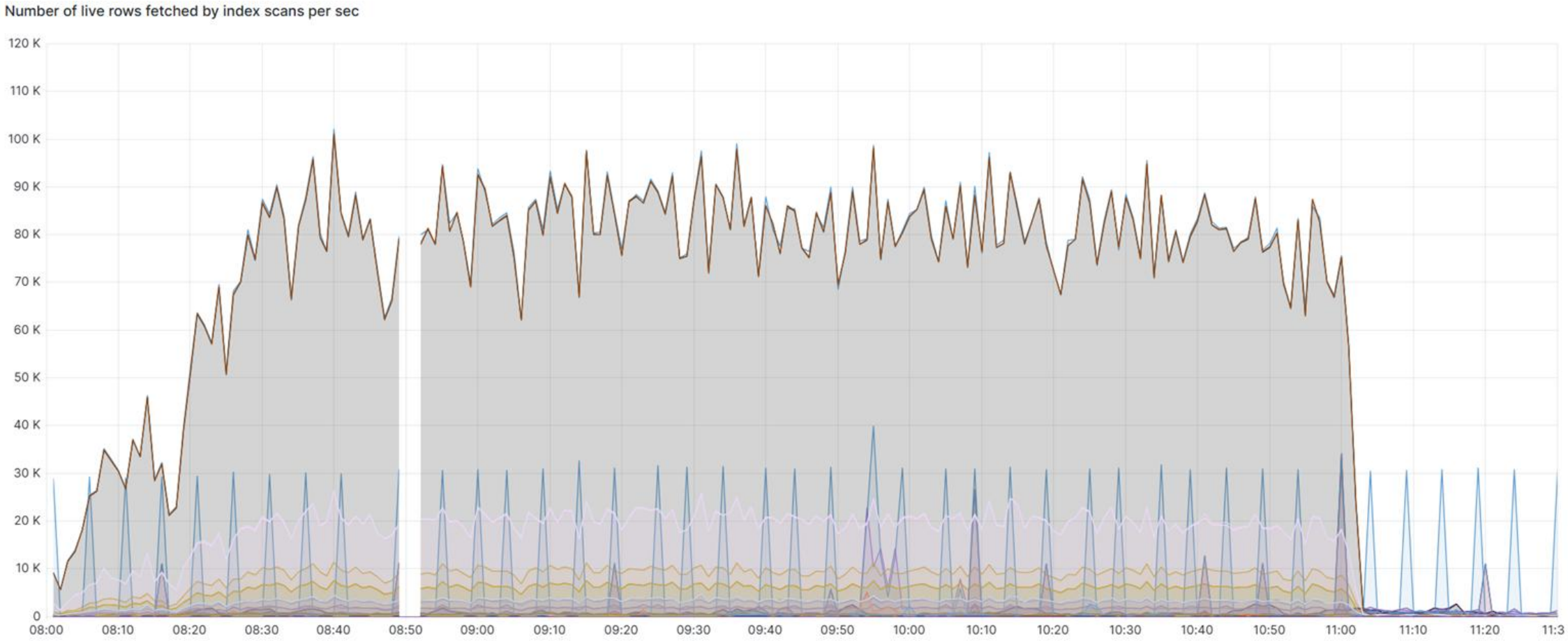
- В отдельных сценариях
- У всех пользователей сразу



PostgreSQL во время инцидентов

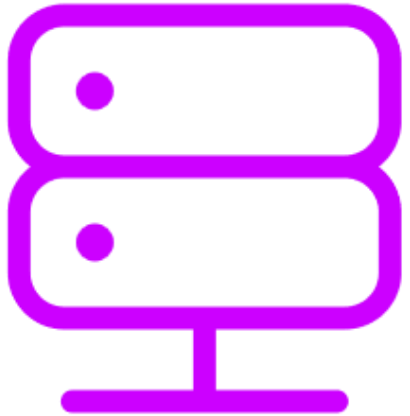


Индексы во время инцидентов

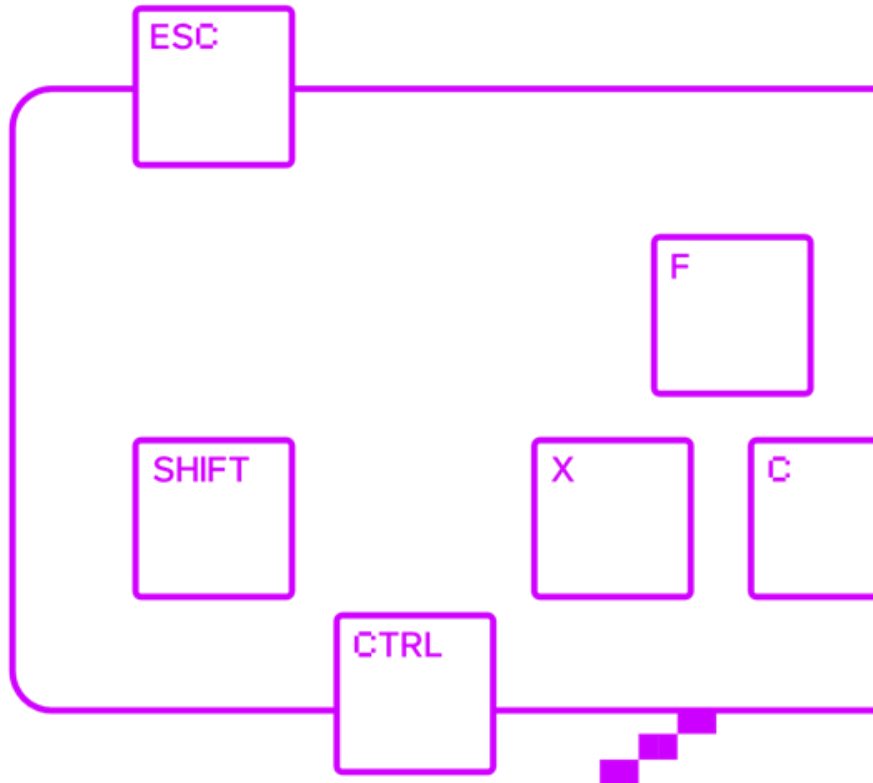


Анализ

- Проблема в работе с PostgreSQL
- Какой запрос тормозит:
  - В конкретном случае
  - Массово
- Проблема не только в медленных запросах

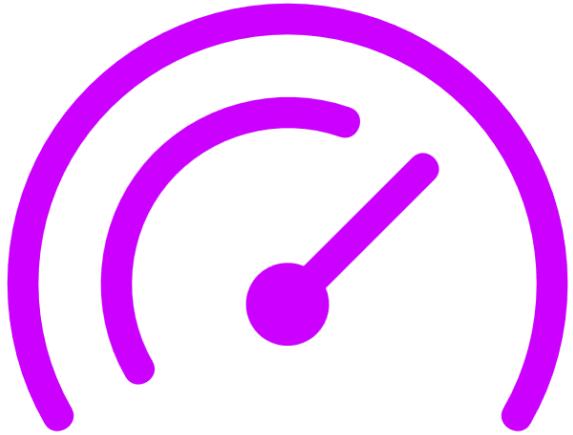


Поиск решения



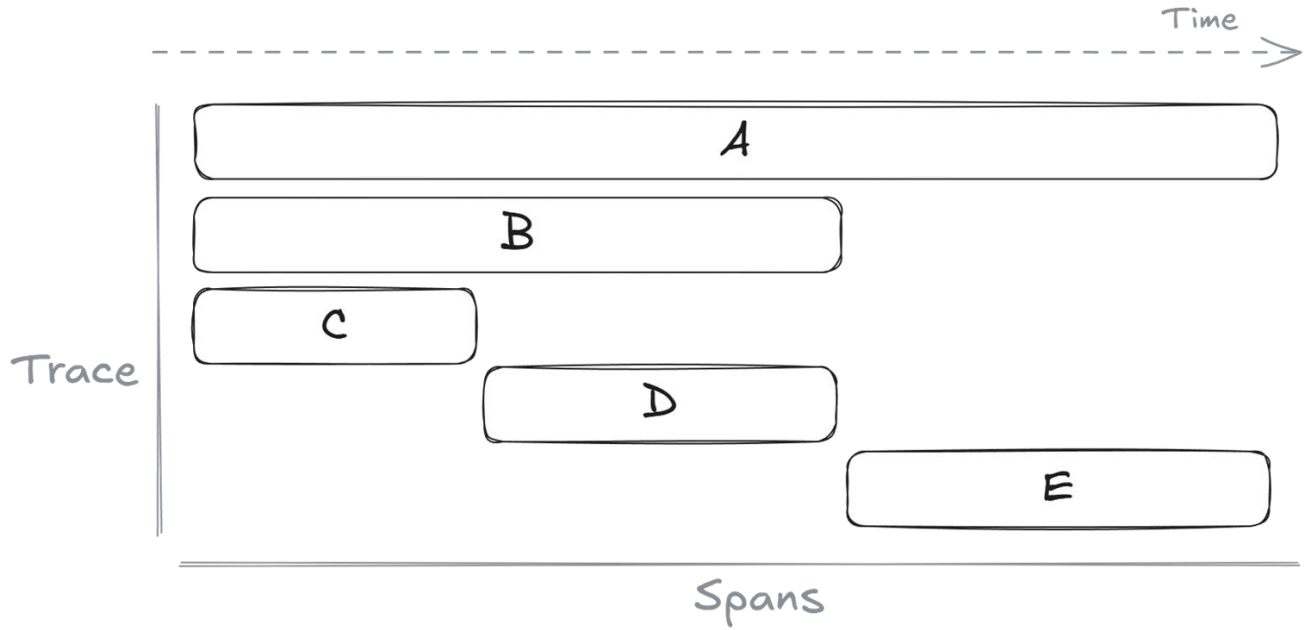
Телеметрия

- Структурированные данные
- Логи
- Метрики
- Распределенные трассировки



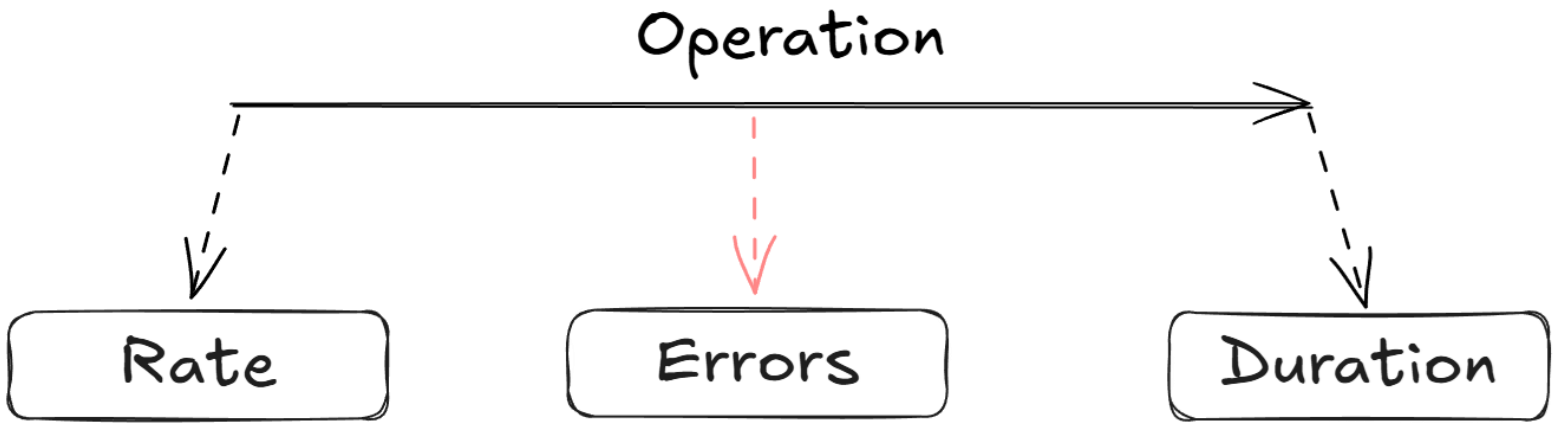
Распределенные трассировки

- Состоят из спанов
- Спаны имеют начало и конец
- Содержат текстовые атрибуты



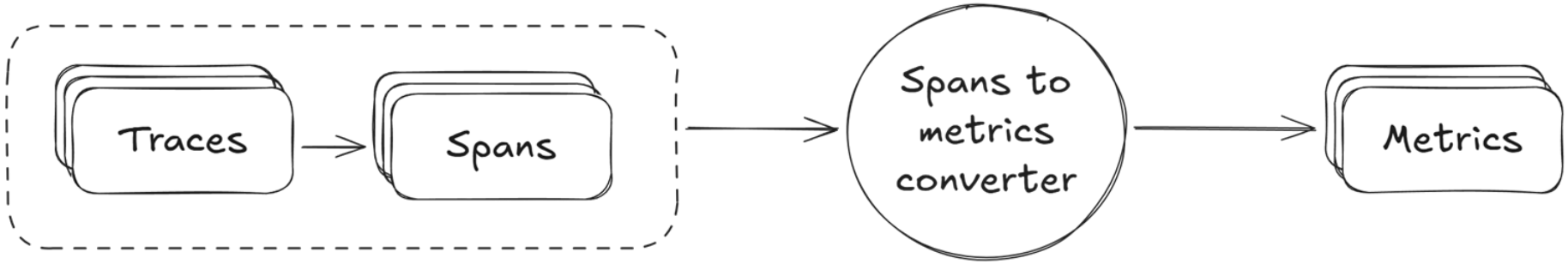
R.E.D Метрики

- Rate — количество запросов
- Errors — количество ошибок
- Duration — длительность



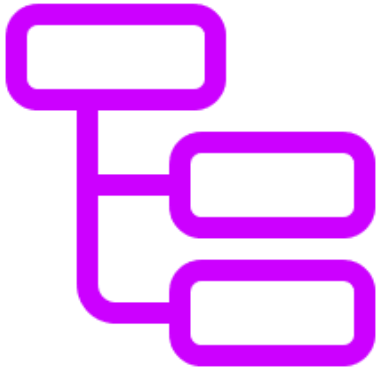
R.E.D метрики из трассировок

- Количество спанов в Rate
- Спаны с ошибками в Error
- Длительность спанов в Duration

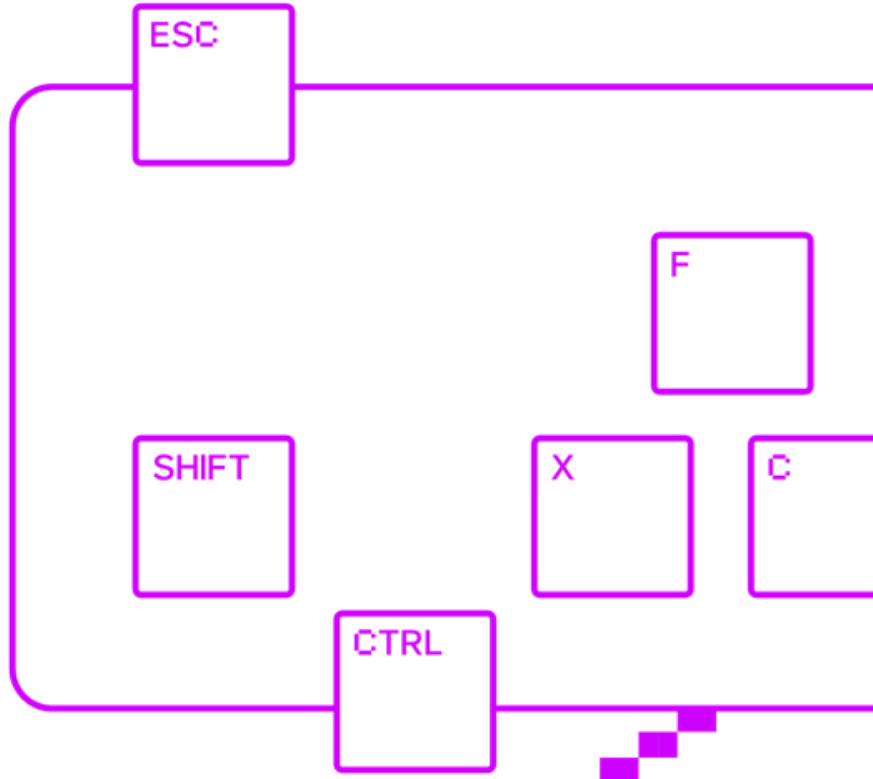


Трассировки обращений к БД

- Спан на обращение к БД
- Имена спанов из кода
- Автоматическое создание спанов



Изменения в приложении



Интерсепторы в EF Core

- Перехватчики операций
- Могут менять объект перехвата
- Требуют явной регистрации

C# Copy

```
public class TaggedQueryCommandInterceptor : DbCommandInterceptor
{
    public override InterceptionResult<DbDataReader> ReaderExecuting(
        DbCommand command,
        CommandEventData eventData,
        InterceptionResult<DbDataReader> result)
    {
        ManipulateCommand(command);

        return result;
    }

    public override ValueTask<InterceptionResult<DbDataReader>> ReaderExecutingAsync(
        DbCommand command,
        CommandEventData eventData,
        InterceptionResult<DbDataReader> result,
        CancellationToken cancellationToken = default)
    {
        ManipulateCommand(command);

        return new ValueTask<InterceptionResult<DbDataReader>>(result);
    }

    private static void ManipulateCommand(DbCommand command)
    {
        if (command.CommandText.StartsWith("-- Use hint: robust plan", StringComparison.Ordinal))
        {
            command.CommandText += " OPTION (ROBUST PLAN)";
        }
    }
}
```

System.Diagnostics.DiagnosticSource

- Внутрипроцессное логирование
- Объекты как полезная нагрузка
- Не нужно менять слой доступа к данным
- EF Core создает события

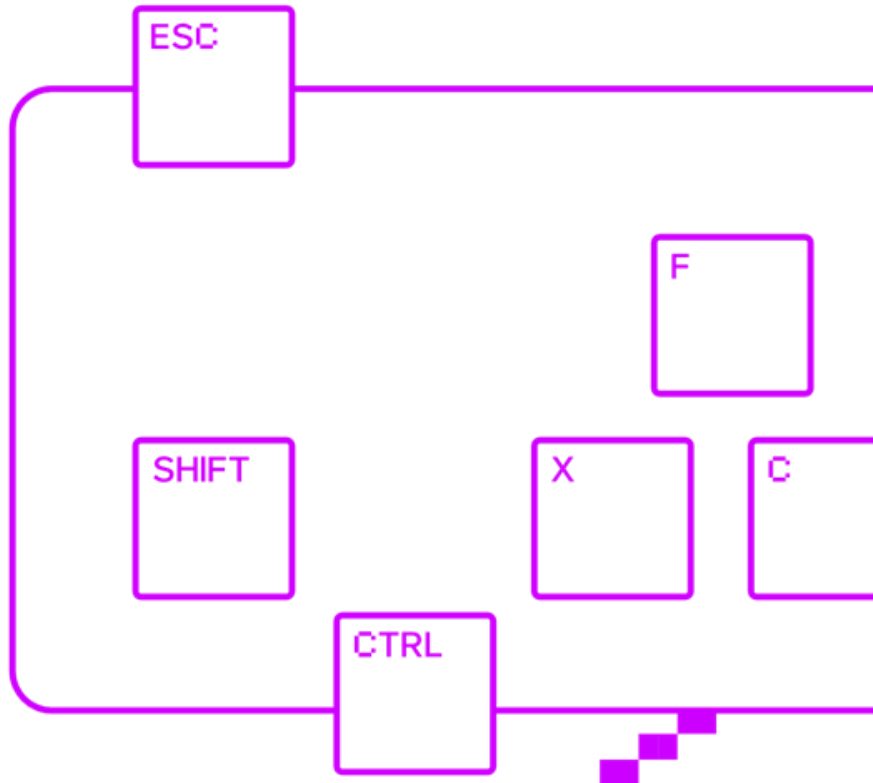
System.Diagnostics.Activity

- Работа со спанами
- AsyncLocal для активного спана
- Можно добавлять атрибуты
- Экспорт спанов из приложения

События от EF Core

- Обработчик событий диагностики
- Начало на создание команды
- Завершение на выполнение

Сбор телеметрии

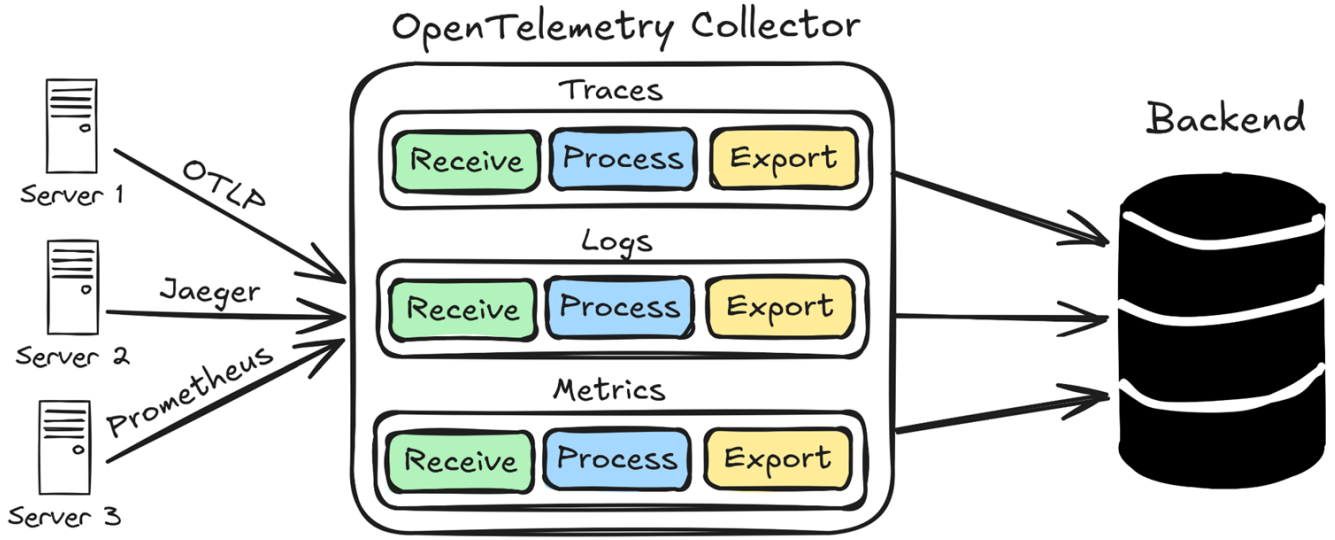


OpenTelemetry

- Открытый фреймворк
- Спецификации и конвенции
- Нативные инструменты для .NET
- Экспорт телеметрии

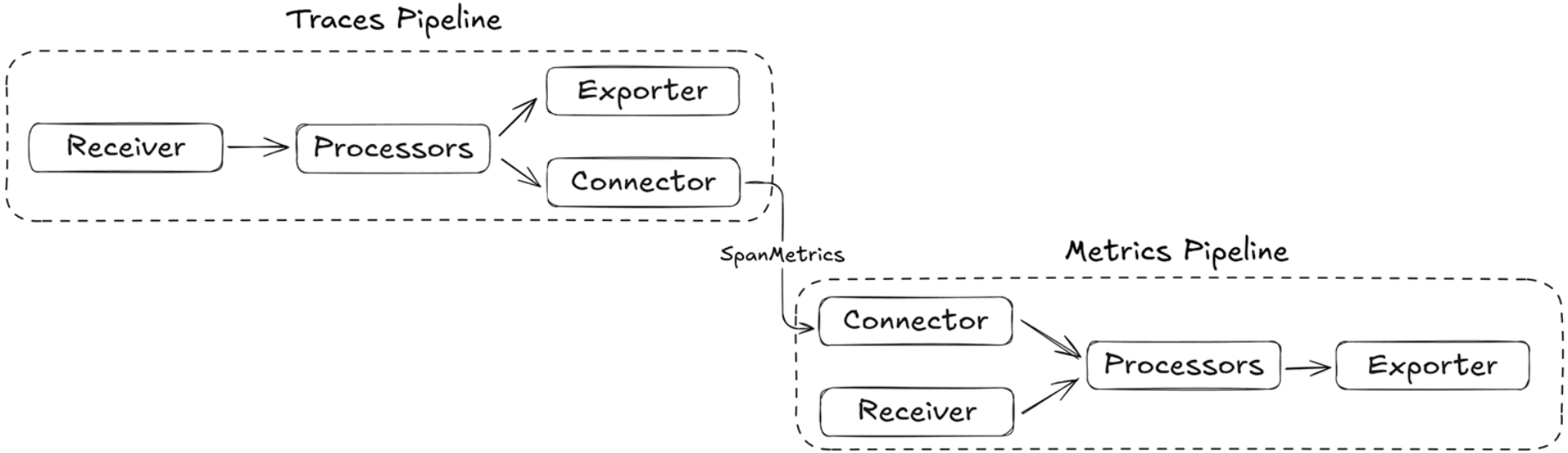
OpenTelemetry collector

- OTLP
- Обработка телеметрии
- Передача в бэкенды



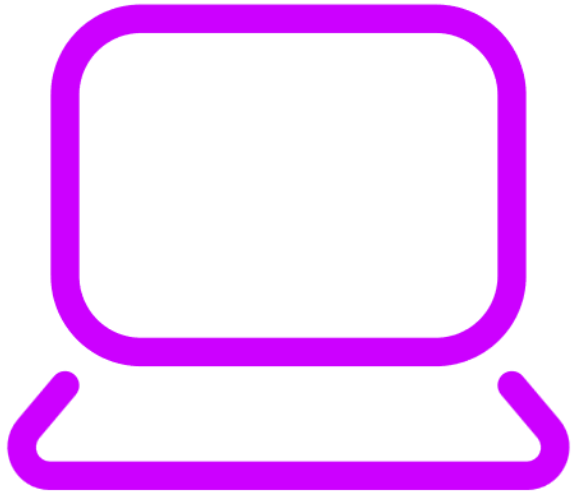
Спаны в метрики

- Span Metrics Connector
- Создает R.E.D метрики
- Разрезы по сервисам и именам спанов

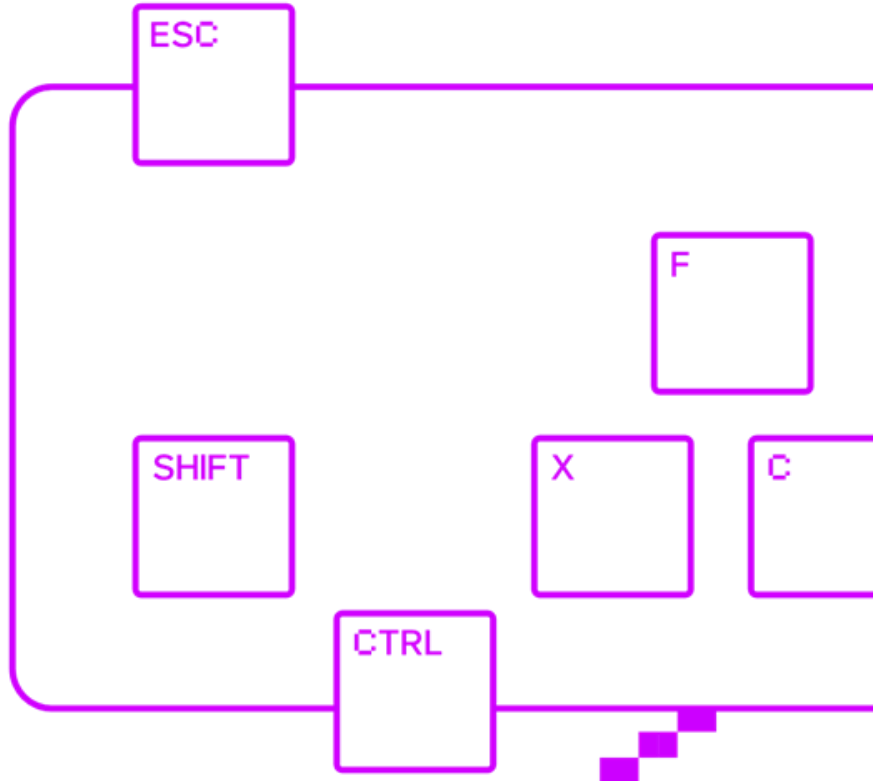


Демонстрация PoC

- ASP.NET Core, EF Core и PostgreSQL
- Prometheus, Grafana, Jaeger
- OpenTelemetry collector

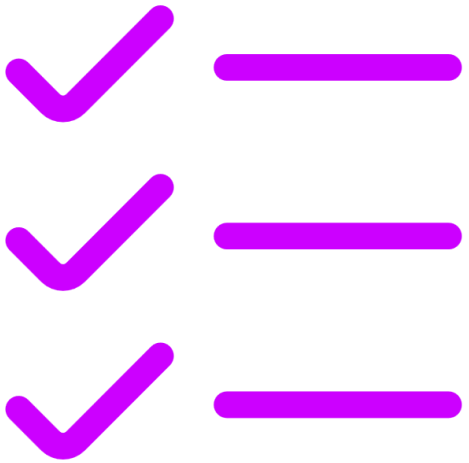


Итоги и ограничения



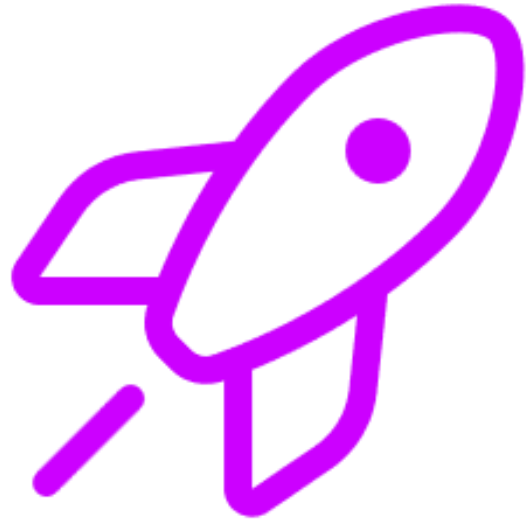
Итог

- Модульное решение
- PoC готового решения
- Настройка процесса мониторинга



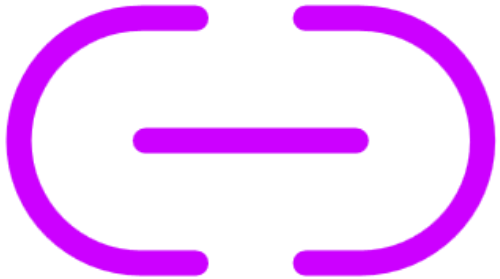
О чем можно еще подумать?

- Сэмплирование спанов
- Количество строк в ответе
- Метрики из самого приложения



Полезные ссылки

- GitHub репозиторий с PoC
- OpenTelemetry .NET Contrib
- OpenTelemetry Docs
- Learning OpenTelemetry by Ted Young, Austin Parker



Подписывайтесь на Технологии в Контуре

Контур