Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.М04-мм

Реализация системы хранения телеметрии для опорной сети 5G

СМИРНОВ Александр Андреевич

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Луцив Д. В.

Консультант:
Эксперт по разработке ПО,
ООО «ЯДРО ЦЕНТР ТЕХНОЛОГИЙ МОБИЛЬНОЙ СВЯЗИ», Иргер А. М.

Санкт-Петербург
2024

Оглавление

Введение

Современные высоконагруженные системы зачастую создаются на основе микросервисной архитектуры, становясь распределёнными [5, 6]. Такая архитектура позволяет добиться хорошей масштабируемости и отказоустойчивости, позволяет удобно внедрять новую функциональность и следовать практикам CI/CD. Подобные системы, как правило, обрабатывают большое количество запросов, при этом в процессе выполнения каждого запроса задействуются несколько узлов системы. Эта особенность усложняет отладку: при обработке запроса каждый узел генерирует собственный набор логов, что затрудняет отслеживание пути этого запроса и сбор информации, касающейся его выполнения. Для решения этой проблемы и облегчения отладки инженерами было введено [9] понятие трейса — сущности, которая представляет путь одного запроса в распределённой системе через её узлы от начала и до конца. Такие трейсы система может генерировать в дополнение к логам. В совокупности логи, трейсы и другая информация, которая позволяет анализировать поведение системы, называются телеметрией [9].

На микросервисной архитектуре, в том числе, построена система ядра опорной сети 5G. Система изображена на рисунке 1. Она представляет собой распределённый набор сервисов, которые общаются между собой посредством REST API. Система постоянно обрабатывает множество разных запросов, которые поступают от базовых станций и других компонентов сети. В терминах сети 5G такие запросы называются процедурами. При выполнении процедур может создаваться различная метаинформация, например, уникальный идентификатор мобильного устройства. Понятие трейса можно удобно отобразить на понятие процедуры, что позволяет применить уже разработанные стандарты для генерации системой телеметрии.

Предметная область телеметрии последние несколько лет активно развивается. На рынке представлены [7] различные решения, которые облегчают работу с такими данными. Однако эти решения рассчитаны на широкий круг пользователей, из-за чего вынуждены абстрагировать-

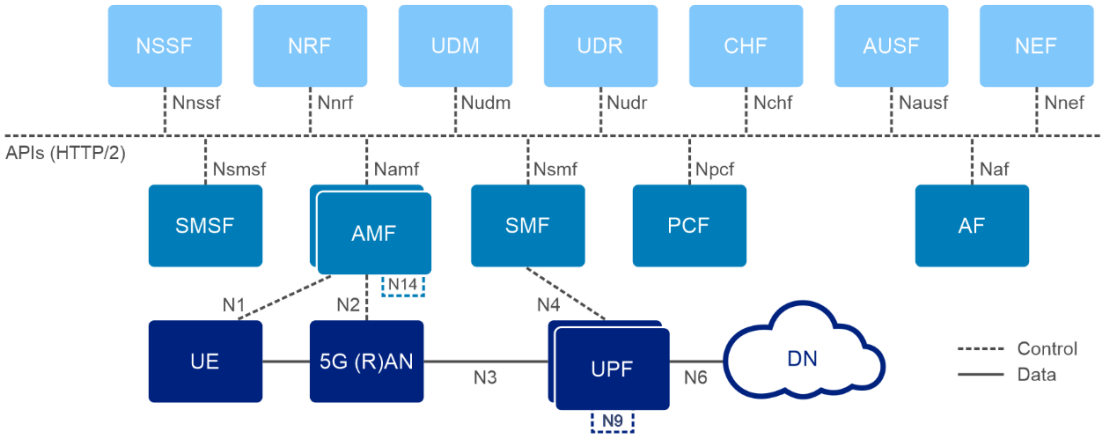

Рис. 1: Архитектура опорной сети 5G
Источник: https://infohub.delltechnologies.com/p/the-5g-core-network-demystified

ся от какой-либо предметной области. Таким образом, было принято решение разработать собственный инструмент для сбора, хранения и обработки телеметрии, ориентированный на специфику предметной области телекоммуникационных данных.

1. Постановка задачи

Целью работы является разработка системы для сбора, хранения и обработки телеметрии опорной сети 5G. Для её выполнения были поставлены следующие задачи:

1. выполнить обзор предметной области в телеметрии;
2. выявить и сформулировать требования к системе;
3. выбрать технологии и разработать архитектуру системы;
4. реализовать прототип системы;
5. произвести апробацию работы системы совместно с ядром опорной сети.

2. Обзор

2.1. Observability: основные понятия

Введём ряд определений, необходимых для понимания предметной области.

Определение 1. ***Observablity*** *(наблюдаемость) — в терминологии распределённых систем это возможность понимать внутренее состояние системы на основании её выходных данных. [9, 3]*

Определение 2. *Выходные данные системы, содержащие информацию о её внутреннем состоянии, называют [9] телеметрией.*

Телеметрия существенно упрощает работу как при отладке системы, так и при поиске и устрании проблем в уже работающей системе. Позволяет ответить на вопрос “Почему это происходит в системе?”.

Телеметрия обычно состоит из трёх видов данных: трейсов, метрик и логов. Рассмотрим каждый из них подробнее.

Лог — некоторое сообщение [11] от системы, содержащее временную метку. Это один из первых и наиболее популярных видов телеметрии. Однако в логах, как правило, не содержится информация о контексте, в котором они были сгенерированы, из-за чего их нельзя связать с конкретным запросом к системе. Поэтому если в распределённой системе проблема возникла в одном из компонентов из-за упущенной проблемы в другом, то такую цепочку становится тяжело распутать.

Трейс — это некоторая совокупность [11] информации о запросе, который обрабатывается системой. В отличие от логов, трейсы содержат информацию о том, как именно конкретный запрос выполнялся внутри системы от начала и до конца (например, REST-запрос на получение какого-то ресурса). Трейс содержит информацию о том, какие операции над запросом были выполнены, через какие узлы системы (например, компоненты облачного приложения) он прошёл.

Спан — это составляющая [11] трейса. Спан представляет информацию об некоторой операции, которая была выполнена при обработке

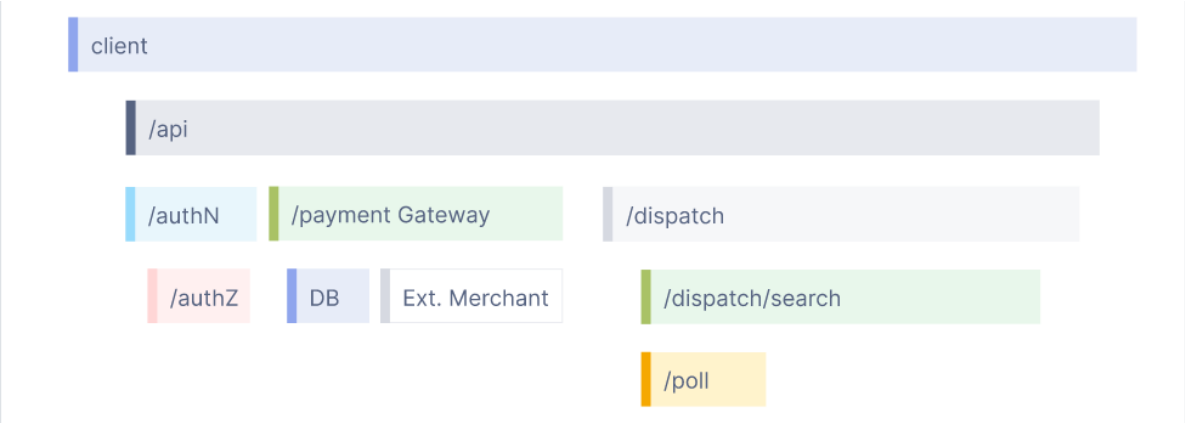

Рис. 2: Пример трейса
Источник: https://opentelemetry.io/docs/concepts/observability-primer

системой запроса (например, запрос системы к базе данных в рамках выполнения REST-запроса). Спан содержит время начала операции и её продолжительность, а также различные метаданные, характеризующие операцию. Также спан содержит информацию о результате выполнения (успех/ошибка).

По своей сути трейс представляет собой совокупность всех спанов, относящихся к одному запросу. Пример трейса изображён на рисунке 2. Минимальный трейс состоит из одного спана, называемного коренным. Коренной спан может быть только один, он представляет весь запрос от начала до конца. Примером коренного спана может служить спан, представляющий вызов коллбэка onHttpRequest.

В случае опорной сети 5G с помощью трейсов удобно представлять процедуры. Процедура в терминах опорной сети — это запрос к системе, затрагивающий несколько её компонент. Пример такого запроса — процедура регистрации телефона в сети. От базовой станции поступает запрос на регистрацию мобильного устройства, после чего он проходит через несколько узлов сети, выполняет нужные действия и завершается. Благодаря трейсам и содержащейся в спанах метаинформации визуализация такого запроса весьма полезна при отладке, а также поиске проблем в процессе эксплуатации.

2.2. Обзор существующих решений

С ростом популярности телеметрии появилось множество observability-решений от разных компаний, предоставляющих собственные инструменты и форматы данных телеметрии. Из-за желания стандартизировать такой подход были созданы проекты OpenTracing [12] и OpenCensus [8], которые впоследствие объединились, в результате чего был создал фреймворк OpenTelemetry.

OpenTelemetry [9] — это открытый фреймворк для обеспечения наблюдаемости в приложениях или системах. Стандарт содержит спецификации для различных типов данных телеметрии (трейсов, метрик и логов), описывает протоколы передачи этих данных, подходы к инструментации кода (иными словами, как писать код, генерирующий телеметрию), различные библиотеки и многое другое.

Одна из важных особенностей стандарта заключается в том, что он позиционируется как vendor-agnostic, то есть он не зависит ни от какой компании. Наоборот, как проприетарные, так и open-source observability-инструменты поддерживают этот стандарт. Это даёт возможность разработчикам приложения единожды инструментировать код, а после чего уже выбирать инструмент под свои нужды.

Рассмотрим одну из проблем, которую решает стандарт. С ростом количества observability-решений от разных вендоров росло и количество спецификаций и протоколов, по которым передаётся телеметрия. Кроме того, для для хранения телеметрии могут использоваться разные хранилища (например, разные базы данных). Поэтому для того, чтобы сохранять телеметрию в нужной базе данных, нужно написать собственный парсер используемого протокола и собственное приложение, сохраняющее данные в базу с нужной схемой. Для решения этой проблемы фреймворк OpenTelemetry предоставляет приложение, называемое OpenTelemetry Collector [10]. Приложение представляет собой прокси, который позволяет принимать телеметрию в разных форматах, изменять её и конвертировать в другие форматы. Конфигурация таких преобразований настраивается с помощью yaml-фала и не требует на-

писания какого-либо кода. Приложение написано на Go, исходный код открыт. Важно отметить, что Collector нигде не хранит телеметрию, а выступает только промежуточным узлом при её обработке.

Хранением, обработкой и визуализацией данных занимаются бэкенды телеметрии. Именно в них OpenTelemetry Collector обычно направляет данные. Как правило, бэкенд состоит из хранилища телеметрии (это может быть база данных или объектное хранилище) и веб-интерфейса, на котором телеметрия отображается (например, в виде графиков). Бэкенды не поставляются фреймворком OpenTelemetry, и каждый вендор реализует их по-своему, предлагая различную функциональность. Однако такие бэкенды часто поддерживают стандарт OpenTelemetry и способны принимать телеметрию в соответствующем стандарту формате. На данный момент существует множество различных как проприетарных, так и open-source бэкендов. Рассмотрим некоторые из них.

2.2.1. Jaeger

Jaeger [4] — один из первых и наиболее известных инструментов для обеспечения наблюдаемости. Инструмент написан на Go, исходный код открыт и доступен под лицезии Apache-2.0. Инструмент собирает спаны и хранит их в базе данных. Также он позволяет визуализировать трейсы и спаны в удобном графическом интерфейсе, который показан на рисунке 3. Однако Jaeger не поддерживает хранение и визуализацию другого типа данных телеметрии —логов. Таким образом, инструмент удобен для быстрого развёртывания и поиска проблем с помощью трейсов, но проигрывает по функциональности более современным инструментам, о которых пойдёт речь далее.

2.2.2. Grafana observability stack

Набор инструментов от компании Grafana Labs [2]. Состоит из четырёх отдельных продуктов.

- Grafana — веб приложение, представляющее собой инструмент

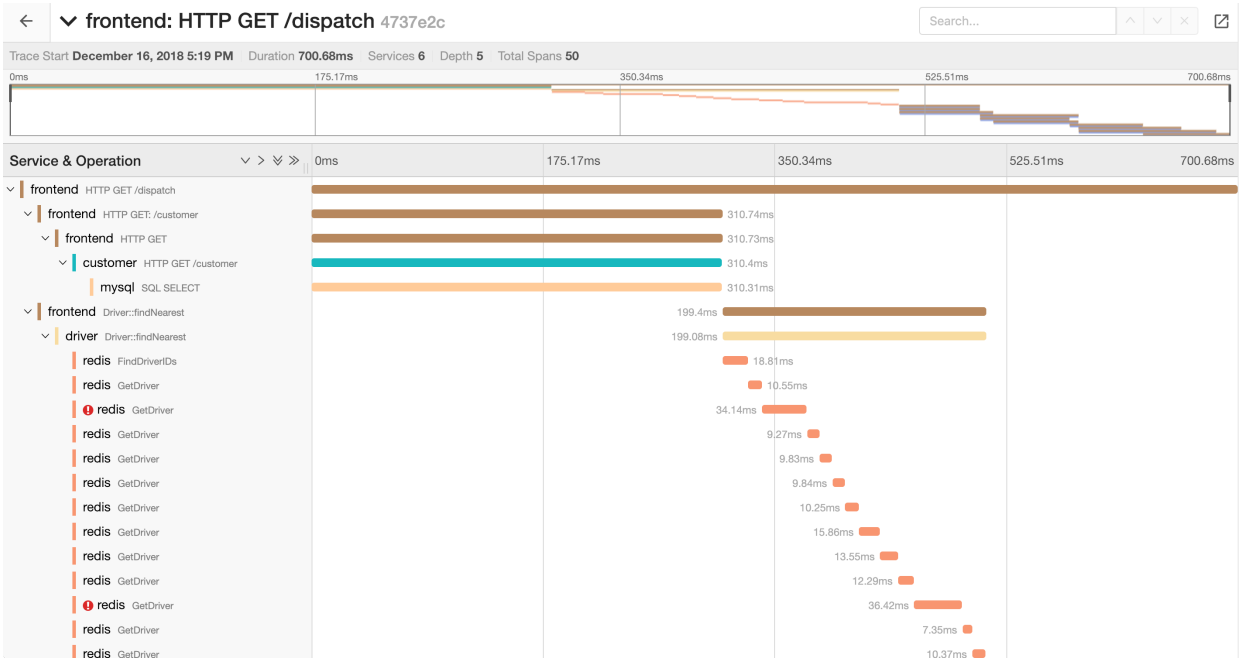

Рис. 3: Пользовательский интерфейс Jaeger-a
Источник: https://www.jaegertracing.io/docs/1.18/frontend-ui

для визуализации различных данных. Тесно используется с time-series базами данных (например, Prometheus) и позволяет с помощью графиков и диаграмм в режиме он-лайн визуализировать метрики. Позволяет создавать ”приборные панели” (dashboards), на которых можно разместить нужные элементы визуализации. Также поддерживает визуализацию трейсов, аналогично интерфейсу Jaeger, и визуализацию логов.

- Grafana Tempo – бэкенд для обработки телеметрии. Инструмент собирает телеметрию от различных узлов и хранит её. Для визуализации данных интегрируется с Grafana.

- Grafana Loki – бэкенд для обработки логов. Собирает и хранит логи.

- Grafana Mimir – инструмент, реализующий долговременное хранилище метрик для БД Prometheus.

В связке эти инструменты представляют мощную систему для мониторинга и обеспечения наблюдаемости. Эта система позволяет собирать, хранить и визуализировать все типы даннных телеметрии. Кроме того,

она даёт возможность коррелировать различные виды телеметрии, относящиеся к одному контексту: например, показать все логи, которые были сгенерированы во время выполнения определённого трейса.

Каждый из продуктов Grafana stack распространяется в двух версиях. Первая версия продуктов имеет открытый исходный код, доступный по лицензии AGPLv3. Вторая версия распространяется под Enterprise лицензией, исходный код закрыт.

Стоит отметить, что инструмент визуализации Grafana — вполне самостоятельная и мощная платформа. Она не ограничивает пользователей использованием Grafana Stack, позволяет визуализировать данные из различных источников. Так, например, она может визуализировать данные напрямую из различных БД, а также сторонних систем сбора трейсов и логов.

2.2.3. SigNoz

SigNoz [13] — тоже система обеспечения наблюдаемости с открытым исходным кодом, написана на Go. Большая часть кода распространяется под лицензией Apache-2.0, однако некоторые компоненты доступны только с Enterprise лицензией. В отличие от Grafana Stack, система обрабатывает все типы данных одним инструментом, за счёт чего она более компактна и проста в развертывании. Систему можно развернуть как локально, так и воспользоваться облачным сервисом, который предоставляет компания-разработчик. Система поддерживает визуализацию трейсов, метрик и логов, создание "приборных панелей" с различной информацией. Данные телеметрии хранятся в базе данных Clickhouse [1], куда их записывает особым образом сконфигурированный OpenTelemetry Collector. Благодаря Clickhouse система позволят выполнять различные сложные запросы к хранящимся данным: например, фильтрация спанов от конкретного сервиса, группировка по атрибутам и агрегирование.

2.3. Выводы

Существующие инструменты, в особенности Grafana Stack и Signoz, предлагают богатую функциональность в области хранения и обработки телеметрии. Однако полная функциональность этих двух инструментов доступна только по Enterprize лицензии. Кроме того, все инструменты ориентированы на работу с широким кругом разнообразных облачных приложений и, соответственно, не имеют привязки к области телекоммуникационных систем. Таким образом, было принято решение реализовать собственный инструмент.

3. Описание решения

3.1. Первоначальные требования

Перед началом работы над системой была организована коммуникация с менеджерами продукта и лидерами команд, в результате чего удалось выявить и сформулировать требования.

3.1.1. Функциональные требования

Для удобства функциональные требования разбиты на три группы. В первой группе находятся требования, касающиеся сбора телеметрии, во второй – касающиеся хранения телеметрии, и в третей — касающиеся её обработки.

1. Сбор телеметрии
    (a) Система по сети принимает телеметрию от различных сетевых функций и сохраняет её в хранилище.
    (b) При получении данных телеметрии система удаляет всю приватную информацию, содержащуюся в них (например, ключи шифрования).
2. Хранение телеметрии
    (a) Система хранит данные в базе данных (БД).
    (b) Система удаляет устаревшие данные для экономии места в БД.
    (c) Система выявляет “интересные” данные и продлевает срок их хранения в БД.
3. Обработка и выдача телеметрии
    (a) Система реализует REST API интерфейс, с помощью которого клиенты могут запрашивать хранящиеся данные.

(b) Система умеет выдавать разные виды телеметрии, относящиеся к одному контексту (например, выводить все логи, относящиеся к конкретному трейсу).

(c) Система умеет находить всю телеметрию с заданным атрибутом (например, все данные по конкретному мобильному устройству).

(d) Система умеет отвечать на запросы, связанные с агрегацией данных по времени (например, среднее время выполнения процедур).

(e) Система поддерживает обработку хранящихся данных посредством пользовательских скриптов.

(f) Система предоставляет web-интерфейс для отображения телеметрии.

3.2. Нефункциональные требования

1. Система должна обрабатывать достаточно большой поток входных данных.

2. Система не обязана быть высокодоступной.

3. Скорость выполнения пользовательских запросов должна быть приемлемой для OLAP-сценариев.

Заключение

В ходе работы были выполнены следующие задачи:

- выполнен краткий обзор предметной области телеметрии: даны основные понятия, выполнен обзор популярных технологий;
- выявлены и сформулированы функциональные и нефункциональные требования к будущей системе.

В следующих семестрах планируется выбрать технологии, реализовать прототип системы и произвести апробацию.

Список литературы

[1] ClickHouse: Fast Open-Source OLAP DBMS. — URL: https://clickhouse.com/.

[2] Grafana. — URL: https://grafana.com/docs/grafana/latest/introduction/.

[3] Grafana — What is observability? — URL: https://grafana.com/docs/grafana-cloud/introduction/what-is-observability/.

[4] Jaeger: open source, distributed tracing platform. — URL: https://www.jaegertracing.io/.

[5] Microservices Adoption in 2020. — URL: https://www.oreilly.com/radar/microservices-adoption-in-2020/.

[6] Microservices architecture. — URL: https://www.atlassian.com/microservices/microservices-architecture.

[7] Observability tools. — URL: https://signoz.io/blog/observability-tools/.

[8] OpenCensus. — URL: https://opencensus.io/.

[9] OpenTelemetry. — URL: https://opentelemetry.io/docs/what-is-opentelemetry/.

[10] OpenTelemetry Collector. — URL: https://opentelemetry.io/docs/collector/.

[11] OpenTelemetry basic concepts. — URL: https://opentelemetry.io/docs/concepts/observability-primer/.

[12] The Opentracing project. — URL: https://opentracing.io/.

[13] SigNoz: Open-Source Observability. — URL: https://signoz.io/docs/.