УДК 004.4

РАЗРАБОТКА ПРОЦЕССОРА ДЛЯ OPENTELEMETRY

Петряков П.В (Университет ИТМО)
Научный руководитель — преподаватель факультета прикладной информатики
Филянин И.В. (Университет ИТМО)

Введение: В современной микросервисной архитектуре, мониторинг и анализ трассировок играют важную роль в обеспечении надежности и производительности. Системы сбора телеметрии, такие как OpenTelemetry, предоставляют стандартизированный подход к сбору данных, но при этом не всегда решают проблему масштабируемости сбора данных трассировки [1]. Статические методы сэмплирования, такие как вероятностное сэмплирование, не позволяют эффективно управлять объемом данных и могут пропускать важные события и аномалии. Для решения этой проблемы необходимо использовать методы динамического сэмплирования, которые позволяют адаптировать сбор данных к текущему состоянию системы [2]. В данном исследовании предлагается подход к динамическому сэмплированию, который использует метрики, собранные с помощью Prometheus и кастомный процессор для OpenTelemetry Collector. Использование Prometheus в качестве источника данных для динамического сэмплирования позволяет эффективно использовать существующую инфраструктуру мониторинга и гибко настраивать критерии принятия решений.

Основная часть. Цель исследования: Разработка методики и создание кастомного процессора для OpenTelemetry Collector, реализующего динамическое сэмплирование на основе данных мониторинга из Prometheus.
В ходе изучения документации OpenTelemetry Collector, технических спецификаций Prometheus и анализа существующих подходов к динамическому сэмплированию, были выявлены ключевые критерии и принципы для разработки процессора динамического сэмплирования, работающего на основе данных из Prometheus:

1. Получение метрик Prometheus через Remote Read API:Для минимизации нагрузки на систему и получения только необходимых данных было принято решение использовать Remote Read API Prometheus. Это позволяет запрашивать у Prometheus только те метрики, которые необходимы процессору для принятия решений о сэмплировании, избегая загрузки всех метрик целиком [3].
2. Анализ метрик с использованием PromQL: Выбор метрик и логика их обработки для динамического сэмплирования основаны на языке PromQL. Выбор PromQL был обусловлен его выразительностью и возможностью агрегировать, фильтровать и трансформировать метрики, предоставляемые Prometheus [4]. Это позволяет выявлять проблемные области системы, используя такие показатели, как:Частота ошибок,Задержки запросов,Нагрузка на сервисы,Специфичные метрики.
3. Критерии динамического сэмплирования:На основе анализа литературы и существующих практик были определены критерии для сэмплирования трасс: на основе ошибок,на основе задержек, на основе нагрузки,Пороговые значения.

Выводы. В результате исследования и разработки был создан кастомный процессор для OpenTelemetry Collector, позволяющий реализовать динамическое сэмплирование на основе метрик, полученных из Prometheus. Этот процессор позволяет гибко настраивать

критерии сэмплирования, такие как частота ошибок, задержки запросов и нагрузка на сервисы, что обеспечивает более эффективный сбор данных трассировки. Интеграция с Prometheus и использование PromQL предоставляют мощный механизм для анализа данных мониторинга и принятия решений о сэмплировании в реальном времени. Разработанный подход позволяет снизить нагрузку на инфраструктуру трассировки, оптимизировать затраты на хранение и анализ данных, а также улучшить качество получаемой информации за счет фокуса на наиболее проблемных областях системы.

Список использованных источников:

1. OpenTelemetry Official Website [Электронный доступ], URL: https://opentelemetry.io/ (дата обращения: 10.01.2025)

2.OpenTelemetry Collector Documentation [Электронный доступ], URL: https://opentelemetry.io/docs/collector/ (дата обращения: 10.01.2025)

3.Prometheus Documentation [Электронный доступ], URL: https://prometheus.io/docs/ (дата обращения: 10.01.2025)

4.PromQL Documentation [Электронный доступ], URL: https://prometheus.io/docs/prometheus/latest/querying/basics/ (дата обращения: 10.01.2025)