

БОРИСОВА А. О., ГРИГОРЬЕВЫХ А. В.
ФОРМИРОВАНИЕ И АНАЛИЗ РАСПРЕДЕЛЕННЫХ ТРАССИРОВОК
МИКРОСЕРВИСНОЙ СИСТЕМЫ ДЛЯ ВЫЯВЛЕНИЯ АНОМАЛЬНОГО
ПОВЕДЕНИЯ

УДК 004.7:004.4:004.8, ГРНТИ 50.41.25

Статья поступила в редакцию 24.09.2026

Формирование и анализ распределенных
трассировок микросервисной системы для
выявления аномального поведения

Formation and analysis of
distributed traces of a microservice
system for anomaly detection

А. О. Борисова, А. В. Григорьевых

A.O. Borisova, A.V. Grigoriev

Ухтинский государственный технический
университет, г. Ухта

Ukhta State Technical University,
Ukhta

В статье рассматривается процесс формирования набора распределенных трассировок микросервисной системы для последующего выявления отклонений от нормального поведения. В качестве экспериментальной среды используется OpenTelemetry Demo, представляющая собой демонстрационное приложение, состоящее из взаимодействующих микросервисов и поддерживающее сбор телеметрических данных. Получение и анализ распределенных трассировок выполняются с использованием системы Jaeger. Для проведения эксперимента сформированы обучающая и тестовая выборки нормальных трассировок, а также отдельный набор трассировок с контролируемой аномалией, создаваемой посредством встроенного сценария отказа сервиса рекламы adFailure. Рассмотрен подход к формированию профиля нормального поведения на основе межсервисных связей и частоты их появления. Выполнено преобразование исходных трассировок в набор числовых признаков, характеризующих структуру, временные параметры и наличие ошибок.

The article considers the process of forming a distributed trace dataset of a microservice system for subsequent detection of deviations from normal behavior. OpenTelemetry Demo, a demonstration application consisting of interacting microservices and supporting telemetry data collection, is used as the experimental environment. Distributed traces are collected and analyzed using Jaeger. Training and test datasets of normal traces are formed, as well as a separate trace dataset containing a controlled anomaly generated by the built-in adFailure scenario. An approach to forming a normal behavior profile based on interservice connections and their occurrence frequency is considered. The original traces are transformed into a set of numerical features characterizing their structure, timing parameters, and errors. The obtained datasets can

Полученные выборки могут быть использованы для дальнейшей разработки и экспериментальной проверки методов выявления аномального поведения микросервисных систем. *be used for further development and experimental evaluation of methods for detecting anomalous behavior in microservice systems.*

Ключевые слова: микросервисная архитектура, распределенная трассировка, OpenTelemetry, Jaeger, аномальное поведение, обнаружение аномалий, телеметрия, межсервисные связи

Keywords: microservice architecture, distributed tracing, OpenTelemetry, Jaeger, anomalous behavior, anomaly detection, telemetry, interservice connections

Введение

Современные информационные системы все чаще строятся на основе микросервисной архитектуры, предполагающей разделение программного комплекса на совокупность относительно независимых компонентов. Каждый микросервис выполняет определенную функцию и взаимодействует с другими компонентами посредством сетевых запросов. Такой подход обеспечивает гибкость разработки и масштабирования программных систем, однако одновременно усложняет наблюдение за их работой и поиск причин возникающих отклонений.

При обработке одного пользовательского запроса в микросервисной системе могут последовательно или параллельно взаимодействовать несколько сервисов. В результате ошибка или увеличение времени выполнения отдельного компонента способны повлиять на всю цепочку обработки запроса. Анализ журналов отдельных сервисов при этом не всегда позволяет получить целостное представление о происходящих в системе процессах.

Одним из средств наблюдения за работой распределенных систем является распределенная трассировка. Трассировка отражает путь выполнения запроса через систему и состоит из отдельных операций – spans, содержащих информацию о выполняемой операции, времени ее начала, продолжительности и взаимосвязях с другими операциями [6, 7]. Представление трассировки в виде совокупности связанных spans позволяет анализировать не только работу отдельных компонентов, но и структуру их взаимодействия.

Для хранения, поиска и анализа распределенных трассировок могут использоваться специализированные программные средства. Одним из них является Jaeger – система распределенной трассировки, позволяющая исследовать выполнение распределенных запросов, анализировать зависимости сервисов и выявлять причины возникновения ошибок и снижения производительности [2].

Особый интерес представляет возможность автоматического выявления аномального поведения микросервисных систем на основе накопленных трассировочных данных [3]. Для разработки и экспериментальной проверки соответствующих методов необходимы данные, содержащие примеры как нормальной работы системы, так и работы при наличии известного отклонения.

Целью настоящей работы является формирование и предварительный анализ набора распределенных трассировок микросервисной системы, включающего нормальное и контролируемое аномальное поведение, а также подготовка полученных данных для последующего применения методов выявления аномалий.

Организация экспериментальной среды

Для проведения эксперимента требовалась распределенная система с большим количеством взаимодействующих микросервисов, позволяющая получать распределенные трассировки. В качестве такой системы была выбрана OpenTelemetry Demo.

OpenTelemetry Demo представляет собой демонстрационное приложение, включающее совокупность взаимодействующих сервисов и предназначенное для демонстрации и исследования возможностей OpenTelemetry [4]. Использование готовой микросервисной среды позволило проводить эксперимент с реальными взаимодействиями компонентов без необходимости предварительной разработки собственной распределенной системы.

Для развертывания экспериментальной среды на компьютере был установлен Docker Desktop, после чего получен исходный код OpenTelemetry Demo [5]. Компоненты системы запускались в отдельных Docker-контейнерах. Для эксперимента использовалась минимальная конфигурация OpenTelemetry Demo, запускаемая предусмотренной проектом командой `make start-minimal`.

После запуска была выполнена проверка работоспособности системы. В Docker Desktop отобразилась запущенная группа контейнеров OpenTelemetry Demo (Рисунок 1).

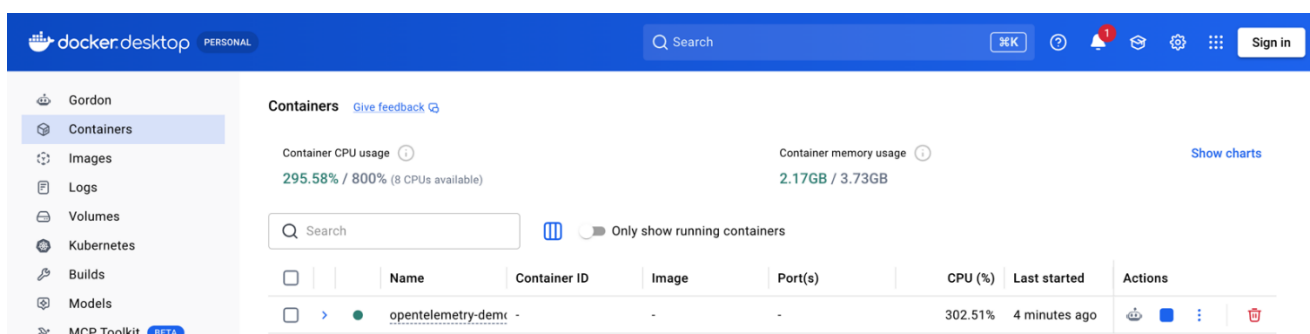


Рисунок 1. Запущенная система OpenTelemetry Demo в Docker Desktop

Для автоматического формирования пользовательских запросов использовался входящий в состав OpenTelemetry Demo компонент `load-generator`. Он имитирует действия пользователей и создает нагрузку на приложение. При

обработке этих запросов микросервисы взаимодействуют между собой, в результате чего формируются распределенные трассировки.

Полученные трассировки передавались в Jaeger. Его веб-интерфейс использовался для первоначального просмотра сформированных трассировок и отдельных операций, входящих в их состав (Рисунок 2).

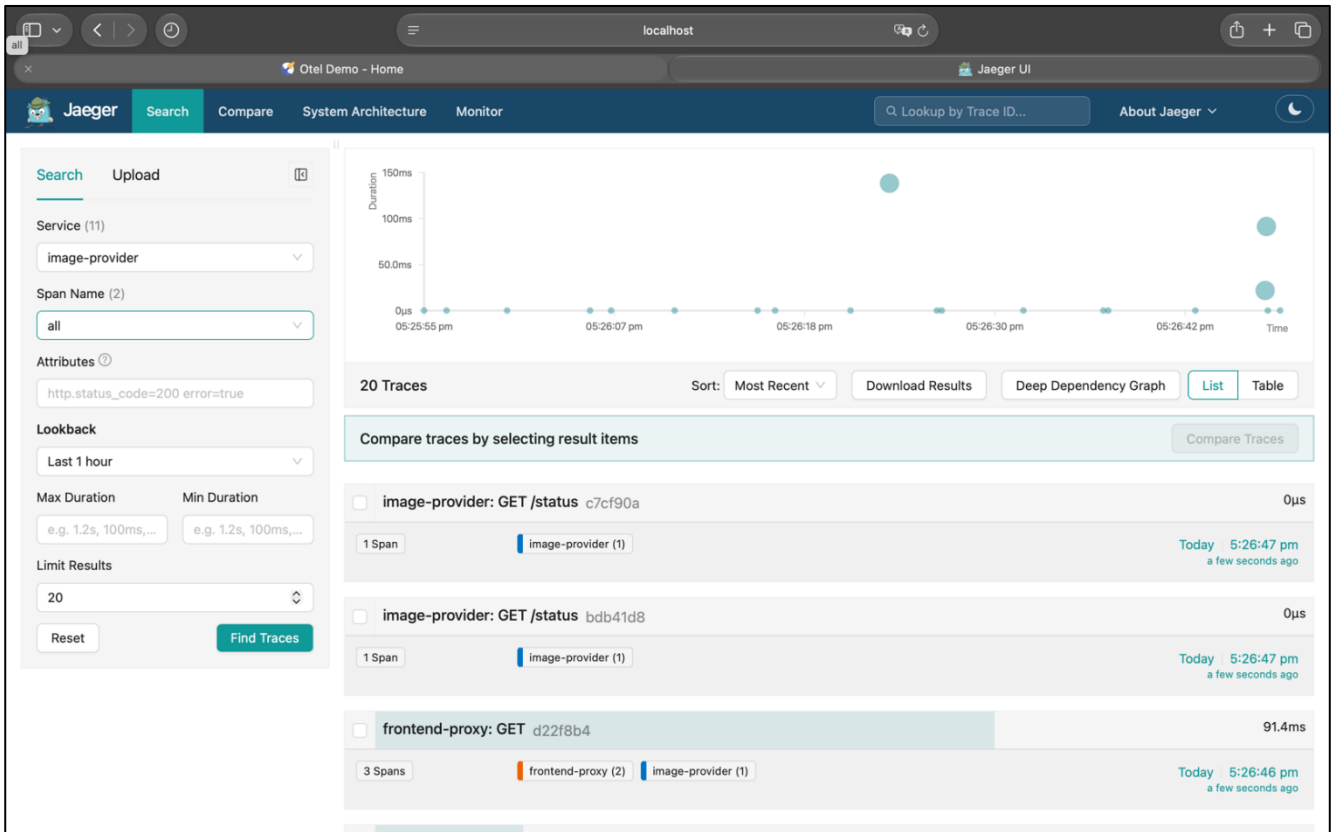


Рисунок 2. Интерфейс Jaeger с полученными распределенными трассировками

Ручная работа через графический интерфейс удобна для изучения отдельных трассировок, однако при формировании экспериментальной выборки требуется обработка значительно большего объема данных. Поэтому дальнейшее получение трассировок выполнялось программно через API Jaeger.

Получение набора нормальных трассировок

На первом этапе через API Jaeger был получен список сервисов, для которых в системе были доступны трассировки. В результате было обнаружено 16 сервисов.

Для предварительной проверки структуры данных было получено несколько трассировок сервиса frontend. Полученные данные сохранялись в формате JSON.

Анализ структуры показал, что трассировка содержит идентификатор traceID, набор отдельных операций spans, названия операций operationName, информацию о связях между операциями references, время начала startTime, продолжительность duration, дополнительные параметры tags, а также

идентификатор процесса processID. Фрагмент структуры полученной распределенной трассировки представлен на Рисунке 3.

Отдельные spans связаны между собой отношениями родительских и дочерних операций. За счет этого трассировка позволяет восстановить последовательность обработки запроса различными компонентами системы. В модели Jaeger трассировка может рассматриваться как ориентированный ациклический граф spans [2].

```
[MacBook-Anastasia ~ % python3 -m json.tool traces.json | head -80
{
  "data": [
    {
      "traceID": "fe2f726c3d6ef44ba292fbb960bb45d7",
      "spans": [
        {
          "traceID": "fe2f726c3d6ef44ba292fbb960bb45d7",
          "spanID": "faf5570f9cf85582",
          "operationName": "router frontend egress",
          "references": [
            {
              "refType": "CHILD_OF",
              "traceID": "fe2f726c3d6ef44ba292fbb960bb45d7",
              "spanID": "f8ec847616dfc209"
            }
          ],
          "startTime": 1789569677884426,
          "duration": 17122,
          "tags": [
            {
              "key": "otel.scope.name",
              "type": "string",
              "value": "envoy"
            },
            {
              "key": "otel.scope.version",
              "type": "string",
              "value": "b579d07d3ad7ee11d32b105e91a5a39ad24718d7/1.39.1/Clean/RELEASE/BoringSSL"
            },
            {
              "key": "component",
              "type": "string",
              "value": "proxy"
            },
            {
              "key": "http.protocol",
              "type": "string",
              "value": "HTTP/1.1"
            },
            {
              "key": "http.status_code",
```

Рисунок 3. Фрагмент структуры распределенной трассировки в формате JSON

После проверки структуры данных был сформирован основной набор нормальных трассировок. Через API Jaeger было запрошено до 1000 трассировок сервиса frontend. Полученные данные были сохранены в отдельный JSON-файл.

В результате был сформирован набор из 1000 распределенных трассировок, содержащих в общей сложности 4878 отдельных операций.

Для последующего проведения эксперимента исходный набор был разделен на обучающую и тестовую выборки. Трассировки случайным образом перемешивались, после чего распределялись между двумя наборами. Для обеспечения воспроизводимости эксперимента использовалось фиксированное

значение `random.seed(42)`, благодаря чему при повторном выполнении разделения сохраняется одинаковое распределение данных.

В результате были сформированы две выборки:

- обучающая выборка (800 нормальных трассировок);
- тестовая выборка (200 нормальных трассировок).

Разделение выполнялось для того, чтобы одни и те же трассировки не использовались одновременно при формировании характеристики нормального поведения и при последующей проверке метода. Таким образом, обучающая выборка предназначена для построения профиля нормального поведения системы, а тестовая – для независимой проверки на ранее не использованных нормальных данных.

Дополнительно была выполнена программная проверка структуры исходного набора. Она показала, что трассировки отличаются количеством входящих в них операций: число spans в отдельных трассировках изменялось от 3 до 110. Следовательно, нормальное функционирование рассматриваемой микросервисной системы характеризуется не единственной фиксированной структурой трассировки, а совокупностью различных вариантов взаимодействия компонентов.

Формирование контролируемой аномалии

Для последующей проверки возможности выявления отклонений требовалось сформировать отдельный набор трассировок, соответствующих заранее известному аномальному состоянию системы.

В OpenTelemetry Demo предусмотрен механизм Feature Flags, позволяющий моделировать различные проблемные сценарии работы отдельных сервисов. Одним из реализованных в проекте сценариев является `adFailure`, предназначенный для генерации ошибок при выполнении операции `GetAds` сервиса рекламы [5].

Данная возможность была использована для создания контролируемой аномалии. Под контролируемой аномалией в настоящей работе понимается преднамеренно создаваемое и заранее известное изменение состояния системы, позволяющее получить трассировки с определенным типом отклонения.

Перед формированием аномальной выборки сценарий `adFailure` находился в выключенном состоянии. Для проведения эксперимента он был активирован, после чего генерация пользовательских запросов продолжалась.

В результате стали формироваться новые распределенные трассировки, содержащие обращения к сервису рекламы и ошибки, возникающие при выполнении соответствующего сценария.

Полученные после активации `adFailure` трассировки были выгружены из Jaeger в отдельный JSON-файл. Для получения данных использовался запрос к API Jaeger с указанием сервиса `ad` и ограничением количества результатов. В результате было получено 257 распределенных трассировок, содержащих spans сервиса рекламы (Рисунок 4).

```

[MacBook-Anastasia ~/opentelemetry-demo % curl -s 'http://localhost:8080/jaeger/ui/api/traces?service=ad&limit=500' > ad_traces_demo.json
[MacBook-Anastasia ~/opentelemetry-demo % python3 -c 'import json; d=json.load(open("ad_traces_demo.json")); print("Получено трассировок:", len(d["data"]))'
Получено трассировок: 257

```

Рисунок 4. Получение трассировок сервиса ad через API Jaeger

Из полученного набора требовалось выделить только трассировки, непосредственно содержащие проявление созданной аномалии. В качестве соответствующего признака использовалось значение `grpc.grpc.status_code = 14`. Согласно спецификации gRPC, код 14 соответствует статусу UNAVAILABLE и означает, что сервис в данный момент недоступен [1].

Для автоматического отбора был разработан скрипт `find_ad_anomalies.py`. Он анализировал сохраненные трассировки, просматривал теги входящих в них spans и выделял трассировки, содержащие заданный код ошибки.

В результате из 257 полученных трассировок было выделено 20 аномальных трассировок. Общее количество spans в них составило 264. Отобранные данные были сохранены в отдельный файл `anomaly_adFailure_demo.json` и использовались в дальнейшем эксперименте.

Такой способ формирования выборки позволяет установить непосредственную связь между изменением состояния микросервисной системы и полученными трассировочными данными. В отличие от поиска неизвестных отклонений в исходном наборе контролируемый сценарий обеспечивает заранее известную причину аномального поведения.

Формирование профиля нормального поведения

Одним из возможных признаков состояния микросервисной системы является структура взаимодействий между ее компонентами. Поэтому для дополнительной характеристики полученных трассировок был проведен анализ межсервисных связей.

Для каждого span устанавливался соответствующий ему сервис. Затем на основании родительских и дочерних отношений между spans определялись пары взаимодействующих сервисов. Каждая такая пара рассматривалась как ориентированная связь графа взаимодействия микросервисов. Полученный для нормальной обучающей выборки граф межсервисных связей представлен на Рисунке 5.

Граф межсервисных связей нормальной обучающей выборки

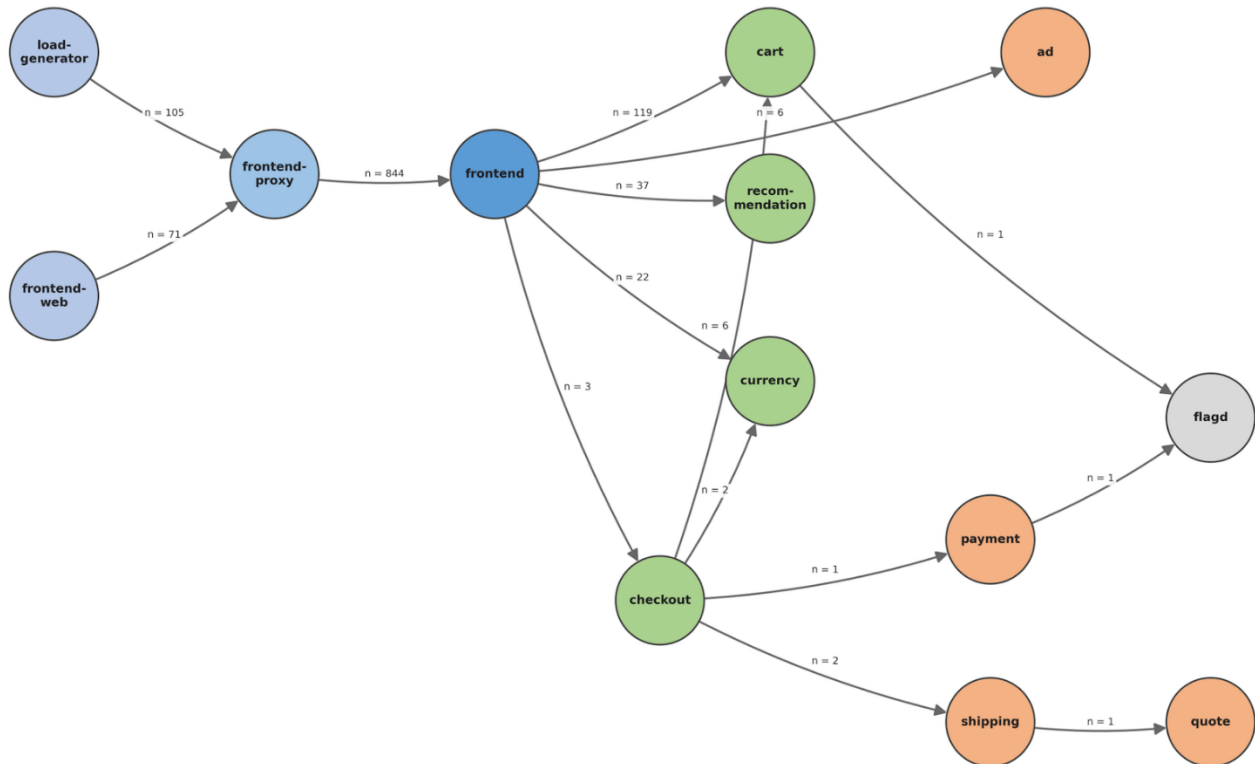


Рисунок 5. Граф межсервисных связей нормальной обучающей выборки

Анализ обучающей выборки из 800 нормальных трассировок показал наличие 15 уникальных межсервисных связей. Наиболее часто встречалась связь frontend-proxy → frontend, зарегистрированная 844 раза. Связь frontend → cart встретила 119 раз, а load-generator → frontend-proxy — 105 раз.

Особый интерес в рамках выбранного сценария представляет связь frontend → ad, соответствующая взаимодействию с сервисом рекламы. В нормальной обучающей выборке она встретила 6 раз.

Для аномального набора было обнаружено только три уникальные межсервисные связи (Рисунок 6). Связи load-generator → frontend-proxy и frontend-proxy → frontend встретились по 40 раз, а взаимодействие frontend → ad — 20 раз.

Полученные результаты показывают, что для характеристики нормального поведения целесообразно учитывать не только факт наличия связи между двумя сервисами, но и частоту ее появления.

Если при последующем анализе обнаруживается межсервисная связь, отсутствовавшая в обучающей выборке, ее появление может рассматриваться как сильный признак отклонения от сформированного профиля. Если связь присутствовала в нормальных данных, но наблюдалась редко, ее появление также может увеличивать степень аномальности, однако в меньшей степени. Напротив, часто встречающиеся при нормальной работе взаимодействия сами по себе практически не свидетельствуют об отклонении.

Таким образом, сформированный профиль нормального поведения содержит информацию о характерных взаимодействиях компонентов микросервисной системы и частоте их появления.

Граф межсервисных связей аномальных трассировок

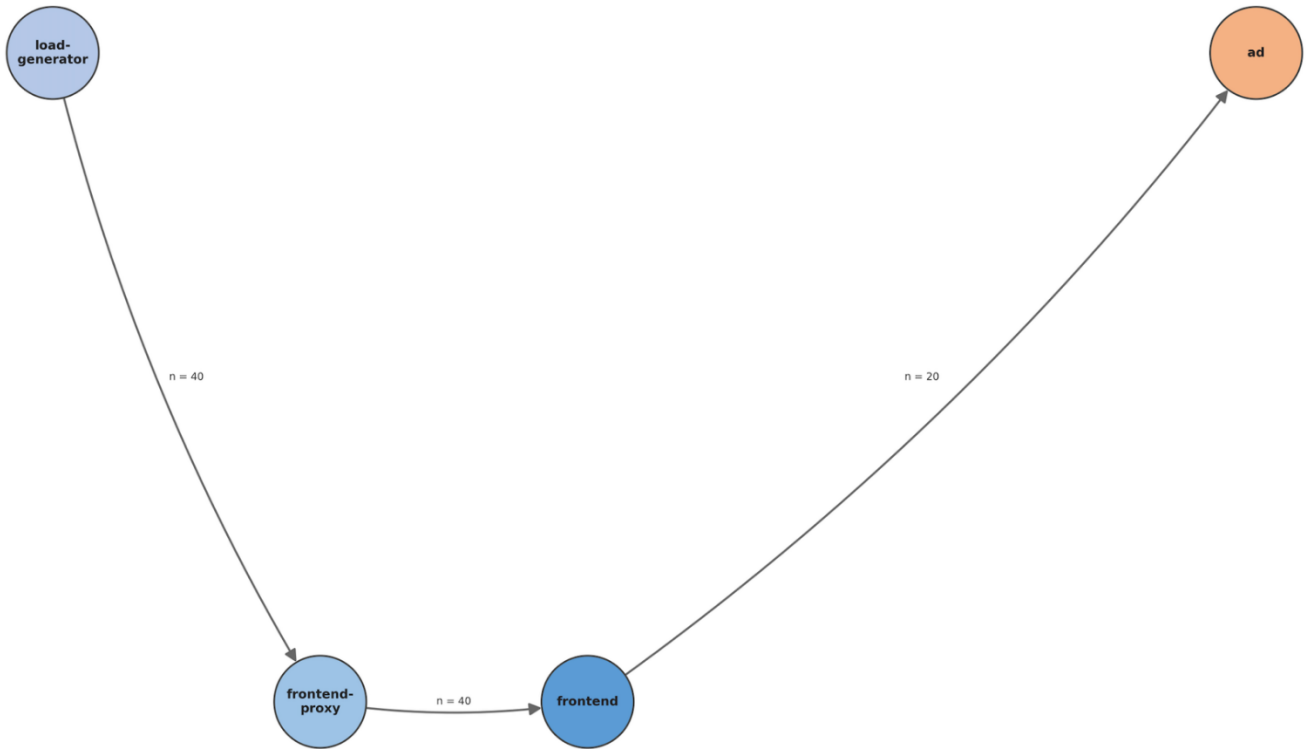


Рисунок 6. Граф межсервисных связей аномальной выборки

Извлечение признаков распределенных трассировок

Исходные данные Jaeger представлены в формате JSON и содержат подробное описание каждой операции. Для дальнейшего количественного анализа каждую распределенную трассировку необходимо представить в виде фиксированного набора числовых характеристик.

Для автоматического извлечения таких характеристик был разработан скрипт `extract_features.py`.

Для каждой трассировки определялись следующие признаки:

- количество spans;
- количество различных сервисов, участвующих в трассировке;
- количество spans с признаком ошибки;
- количество срабатываний сценария `adFailure`;
- общая длительность трассировки;
- средняя длительность span;
- максимальная длительность span.

Для определения сервисов, участвующих в обработке запроса, для каждого span устанавливалось имя соответствующего сервиса на основании связи с процессом, заданной параметром `processID`.

Ошибочными считались операции, для которых значение признака `otel.status_code` соответствовало состоянию `ERROR`.

Дополнительно для аномальных трассировок определялось количество обращений к сервису `ad`, завершившихся с кодом `rpc.grpc.status_code = 14`. Данная характеристика позволяла проверить, что сформированные трассировки действительно содержат проявление созданного в эксперименте сценария отказа.

Обработка выполнялась отдельно для трех сформированных наборов данных. В результате были получены:

- `normal_train_features.csv`: признаки 800 нормальных трассировок обучающей выборки;
- `normal_test_features.csv`: признаки 200 нормальных трассировок тестовой выборки;
- `anomaly_features.csv`: признаки 20 трассировок с контролируемой аномалией.

Фрагмент сформированного набора признаков распределенных трассировок представлен на Рисунке 7. Полученные данные содержат структурные и временные характеристики как нормальных, так и аномальных трассировок, что позволяет использовать их для последующего количественного анализа.

Представление признаков распределённых трассировок
Пример преобразования трассировок в набор числовых признаков

Набор	Трассировка	Spans	Сервисы	Ошибки	Длительность, мс	Средняя длительность span, мс	Макс. длительность span, мс
Обучающая	fb706df7...	14	4	0	126.00	52.75	126.00
Обучающая	e2373948...	3	2	0	9.31	8.48	9.31
Обучающая	3bbc5be1...	9	4	0	299.00	109.43	299.00
Обучающая	912a37af...	6	2	3	62325.77	42227.43	62324.98
Обучающая	358517cb...	3	2	0	25.96	19.36	25.13
Тестовая	b8c2dc54...	9	4	0	220.00	51.37	220.00
Тестовая	a13549bf...	7	3	0	64156.48	38048.97	64152.97
Тестовая	e20d4e89...	9	4	0	47.00	13.25	47.00
Тестовая	e948c34f...	3	2	0	6.64	5.88	6.64
Тестовая	5c2a24cd...	3	2	0	129.09	122.06	129.09
Аномальная	15927e47...	13	4	8	338.16	175.51	338.16
Аномальная	1f499ec6...	13	4	8	611.49	310.88	611.49
Аномальная	fe0522ad...	13	4	8	99.88	47.32	99.88

Рисунок 7. Фрагмент результатов извлечения признаков распределенных трассировок

Таким образом, исходные JSON-данные были преобразованы в структурированный набор числовых характеристик, пригодный для

последующего количественного анализа и расчета степени отклонения отдельных трассировок от нормального поведения.

Заключение

В работе рассмотрен процесс формирования экспериментального набора распределенных трассировок микросервисной системы для последующего исследования методов выявления аномального поведения.

В качестве экспериментальной среды использована OpenTelemetry Demo, позволяющая моделировать работу распределенного приложения и получать телеметрические данные о взаимодействии его компонентов. С использованием Jaeger был сформирован исходный набор из 1000 нормальных распределенных трассировок, содержащих 4878 отдельных операций.

Исходные данные были разделены на обучающую выборку из 800 и тестовую выборку из 200 нормальных трассировок. Такое разделение позволяет независимо формировать профиль нормального поведения и проводить последующую проверку метода.

Для получения данных с известным отклонением использован предусмотренный OpenTelemetry Demo сценарий adFailure. В результате была создана контролируемая аномалия и сформирована отдельная выборка из 20 трассировок, содержащих соответствующий признак ошибки.

Проведенный анализ показал возможность использования структуры межсервисных взаимодействий и частоты возникновения отдельных связей для характеристики нормального поведения микросервисной системы. Дополнительно для каждой трассировки сформирован набор числовых признаков, включающий структурные и временные характеристики, а также сведения об ошибочных операциях.

Полученные обучающая, тестовая и аномальная выборки создают основу для следующего этапа исследования – разработки метода количественной оценки степени отклонения распределенной трассировки от нормального поведения и экспериментальной проверки возможности автоматического обнаружения аномалий в микросервисных системах.

Список использованных источников и литературы:

1. gRPC. Status Codes [Электронный ресурс]. Режим доступа: <https://grpc.io/docs/guides/status-codes/> (дата обращения: 16.09.2026).
2. Jaeger. Documentation [Электронный ресурс]. Режим доступа: <https://www.jaegertracing.io/docs/> (дата обращения: 20.08.2026).
3. Liu P., Xu H., Ouyang Q. et al. Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks // 31st IEEE International Symposium on Software Reliability Engineering (ISSRE). 2020.
4. OpenTelemetry. OpenTelemetry Demo [Электронный ресурс]. Режим доступа: <https://opentelemetry.io/docs/demo/> (дата обращения: 04.08.2026).
5. OpenTelemetry Demo: исходный код проекта [Электронный ресурс]. Режим доступа: <https://github.com/open-telemetry/opentelemetry-demo> (дата обращения: 11.08.2026).

6. Shkuro Y. Mastering Distributed Tracing: Analyzing Performance in Microservices and Complex Systems. Birmingham: Packt Publishing, 2019. 444 p.

7. Sigelman B. H., Barroso L. A., Burrows M. et al. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google, 2010. [Электронный ресурс]. Режим доступа: <https://research.google/pubs/dapper-a-large-scale-distributed-systems-tracing-infrastructure/> (дата обращения: 19.08.2026).

List of References:

1. gRPC. Status Codes [Electronic resource]. Access mode: <https://grpc.io/docs/guides/status-codes/> (date of access: 16.09.2026).

2. Jaeger. Documentation [Electronic resource]. Access mode: <https://www.jaegertracing.io/docs/> (date of access: 20.08.2026).

3. Liu P., Xu H., Ouyang Q. et al. Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks // 31st IEEE International Symposium on Software Reliability Engineering (ISSRE). 2020.

4. OpenTelemetry. OpenTelemetry Demo [Electronic resource]. Access mode: <https://opentelemetry.io/docs/demo/> (date of access: 04.08.2026).

5. OpenTelemetry Demo: Project Source Code [Electronic resource]. Access mode: <https://github.com/open-telemetry/opentelemetry-demo> (date of access: 11.08.2026).

6. Shkuro Y. Mastering Distributed Tracing: Analyzing Performance in Microservices and Complex Systems. Birmingham: Packt Publishing, 2019. 444 p.

7. Sigelman B. H., Barroso L. A., Burrows M. et al. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google, 2010. [Electronic resource]. Access mode: <https://research.google/pubs/dapper-a-large-scale-distributed-systems-tracing-infrastructure/> (date of access: 19.08.2026).