

УДК 004.42:004.774:004.67

Реконструкция межсервисных отношений по трассировочным данным

**Забродин Андрей
Владимирович**

— кандидат ист. наук, доцент. Доцент кафедры «Информационные и вычислительные системы». Научные интересы: информационные системы, аналитика данных, парадигмы программирования, микросервисная архитектура, облачные технологии.
E-mail: zabrodin@pgups.ru

**Хонина Ольга
Игоревна**

— ассистент кафедры «Информационные и вычислительные системы». Научные интересы: информационные системы, веб-разработка, парадигмы программирования.
E-mail: olga.khonina.0311@mail.ru

**Фадхутдинов Кирилл
Эдуардович**

— студент 3-го курса бакалавриата направления 09.03.01 «Информатика и вычислительная техника». Научные интересы: информационные системы, веб-разработка.
E-mail: shiroasuri@gmail.com

Петербургский государственный университет путей сообщения Императора Александра I, Россия, 190031, Санкт-Петербург, Московский пр., 9

Для цитирования: Забродин А. В., Хонина О. И., Фадхутдинов К. Э. Реконструкция межсервисных отношений по трассировочным данным // Интеллектуальные технологии на транспорте. 2026. № 3 (47). С. 63–71. DOI: 10.20295/2413-2527-2026-347-63-71

Аннотация. Рассматривается проблема перехода от анализа отдельной распределенной трассы к характеристикам межсервисных отношений, которые не содержатся в завершенном виде в одном выполнении и выявляются при сопоставлении множества трассировочных наблюдений. **Цель:** формализовать способ реконструкции межсервисных отношений по совокупности трассировочных данных и проверить его реализуемость на программном прототипе. **Методы:** выделение межсервисных вызовов из распределенных трасс, объединение однотипных наблюдений по идентифицирующему ключу, расчет характеристик соответствующих отношений и их представление в ориентированном атрибутированном графе. **Результаты:** отдельные вызовы представлены как реализация межсервисных взаимодействий, а характеристики отношений определяются по совокупности таких реализаций. Предложенная процедура реализована в программном прототипе и проверена на синтетических трассировочных данных микросервисной системы из пяти компонентов в сценариях штатной работы, повышенной задержки и ошибочных вызовов. Эксперимент показал воспроизводимость предусмотренного формализацией соответствия между изменениями отдельных наблюдений и характеристиками реконструируемых межсервисных отношений. **Практическая значимость:** получаемая графовая модель позволяет соотнести характеристики, отсутствующие в завершенном виде в отдельной трассе, с конкретными сервисами и направлениями их взаимодействия.

Ключевые слова: распределенная трассировка, микросервисная архитектура, межсервисные отношения, графовая модель, наблюдаемость, OpenTelemetry, трассировочные данные, системное представление, реконструкция взаимодействий

2.3.5 — математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей (технические науки)

Введение

В распределенной системе поведение запроса распадается на последовательность локальных операций, ни одна из которых не содержит целостного знания о выполнении. Распределенная трассировка преодолевает эту фрагментарность, связывая операции разных сервисов в единую причинно-временную структуру [1, 2]. Однако целостность, которую дает трасса, ограничена рамками одного выполнения. Маршрут запроса, длительность отдельных операций, место возникновения ошибки прослеживаются внутри нее [3], но поведение системы на интервале наблюдения — предмет иного рода: частота взаимодействий, доля ошибок, характер и распределение задержек не принадлежат ни одной отдельной трассе в завершенном виде и обнаруживаются лишь при сопоставлении ряда выполнений.

С переходом к интервалу наблюдения меняется и единица анализа. Отдельный вызов рассматривается уже как реализация межсервисного взаимодействия, повторяющегося от трассы к трассе. Чтобы определить характеристики такого взаимодействия, однотипные вызовы необходимо объединить по признакам, позволяющим отнести их к одному отношению. Характеристики отношения рассчитываются по совокупности зарегистрированных реализаций.

В существующих исследованиях трассировочные данные используются для получения сервисных метрик и графов зависимостей [4], а также для агрегированного анализа множества распределенных трасс [5, 6]. Для данной работы существенно не само агрегирование наблюдений, а то, к какому объекту относятся полученные характеристики. В предлагаемой модели таким объектом служит межсервисное отношение, реконструируемое по совокупности однотипных вызовов.

Реконструировать отношение и определить его характеристики недостаточно: результат необходимо соотнести с элементами системы, участвующими во взаимодействии. Для этого используется ориентированный атрибутированный граф, локализирующий реконструированное знание относительно конкретных сервисов и направлений взаимодействия.

Научная новизна работы состоит в формализации перехода от единичных трассировочных наблюдений к межсервисным отношениям как самостоятельным объектам системного уровня и в представлении их реконструированных характеристик в ориентированной атрибутированной графовой модели. Цель исследования — экспериментально проверить предложенный способ реконструкции. Авторы исходят из гипотезы, что характеристики межсервисного отношения, которых нет в завершенном виде ни в одной трассе, можно реконструировать, сопоставив множество трассировочных наблюдений и соотнеся их с элементами графовой модели. Для проверки реализован программный прототип, переводящий отдельные наблюдения в системное представление, и поставлен эксперимент на синтетических данных микро-сервисной системы.

От единичного выполнения к межсервисному отношению

Переход от анализа отдельной трассы к сопоставлению множества выполнений уже используется в ряде подходов к исследованию распределенных систем. В [4] трассировочные данные рассматриваются как источник сервисных метрик, графов зависимостей и рабочих потоков. В [5] множество распределенных трасс используется для агрегированного визуального анализа. В TraVista агрегированная информация, напротив, служит дополнительным контекстом при интерпретации конкретной трассы [6]. Несмотря на различие задач, эти подходы показывают, что характеристики системного уровня могут быть получены при сопоставлении данных нескольких выполнений.

Для решаемой задачи существенен вопрос о том, к какому объекту относятся характеристики, полученные при таком сопоставлении. Частота вызовов, доля ошибок и характеристики задержек описывают уже не отдельное выполнение, а повторяющееся взаимодействие между сервисами. Таким объектом в предлагаемой модели является межсервисное отношение. Каждый зарегистрированный вызов рассматривается как одна его реализация,

а совокупность однотипных вызовов служит основанием для определения характеристик отношения.

Выделение межсервисного отношения как объекта реконструкции позволяет отделить получение системного представления от его последующей диагностической интерпретации. Прежде чем анализировать задержки, ошибки или другие отклонения, необходимо определить объект, относительно которого они вообще могут быть зафиксированы. Поэтому способ реконструкции отношения не связывается с конкретным методом диагностики: сначала по множеству наблюдений восстанавливаются само отношение и его характеристики, а уже затем они могут использоваться для анализа поведения системы.

Графовое представление межсервисных взаимодействий само по себе не является новым. Например, Kialí на основе телеметрии Istio строит граф наблюдаемого трафика, отображающий интенсивность запросов, задержки и состояние связей между компонентами [7]. В данной работе существенен другой аспект — происхождение представленных в графе ребер и их характеристик. Каждое ребро соответствует межсервисному отношению, реконструированному по множеству однотипных вызовов, а его атрибуты получают значения из совокупности относящихся к этому взаимодействию наблюдений.

Формализация реконструкции межсервисных отношений

Реконструкция межсервисных отношений по трассировочным данным предполагает выделение среди зарегистрированных вызовов тех, которые относятся к одному взаимодействию. Для этого необходимо задать правило такого отнесения и определить характеристики, вычисляемые по совокупности соответствующих наблюдений. Исходной единицей анализа будем считать наблюдение o_i — направленный вызов между двумя сервисами в рамках конкретного распределенного выполнения:

$$o_i = (s_i, d_i, a_i, \tau_i, q_i, \dots),$$

где s_i — сервис-источник;

d_i — сервис-получатель;

a_i — тип операции;

τ_i — задержка;

q_i — статус завершения вызова.

Первые признаки позволяют идентифицировать взаимодействие, остальные описывают его конкретную реализацию.

При выбранном уровне детализации взаимодействия задается ключом

$$K(o_i) = (s_i, d_i, a_i).$$

Если тип операции для решаемой задачи несущественен, используется более общий ключ:

$$K(o_i) = (s_i, d_i).$$

Таким образом, состав ключа определяет, какие вызовы будут рассматриваться как реализации одного межсервисного отношения.

Пусть

$$O = \{o_1, o_2, \dots, o_n\} —$$

множество наблюдений, полученных из выбранного массива трасс. Для каждого значения ключа k выделяется подмножество

$$R_k = \{o_i \in O \mid K(o_i) = k\}.$$

Оно объединяет зарегистрированные реализации одного взаимодействия и служит основанием для реконструкции соответствующего межсервисного отношения.

Набор таких характеристик обозначим как

$$A(R_k) = \langle n_k \mid \text{error_rate}_k \mid \text{latency_stats}_k \mid \dots \rangle,$$

где n_k — число зарегистрированных вызовов;

error_rate_k — доля завершившихся с ошибкой, %;

latency_stats_k — статистические характеристики распределения задержек.

Конкретный состав показателей определяется задачей анализа.

Результат представляется ориентированным атрибутированным графом $G = (V, E)$, где вершины V соответствуют сервисам, а каждое ребро $e_k \in E$ — реконструированному межсервисному отношению. Значения $A(R_k)$ записываются в атрибуты соответствующего ребра. Если тип операции включен в ключ, между одной парой сервисов могут быть представлены несколько различных отношений.

Состав реконструированного графа определяется исходными трассировочными наблюдениями. Временной интервал, набор сервисов и полнота трассировки определяют, какие взаимодействия будут в нем представлены и на основании какого числа реализаций будут рассчитаны их характеристики [8]. Поэтому отсутствие ребра свидетельствует только о том, что соответствующее взаимодействие не наблюдалось в рассматриваемом массиве данных. Оно не означает, что такое взаимодействие в системе невозможно. По этой же причине реконструированный граф не следует отождествлять с архитектурной схемой: последняя задает предусмотренные компоненты и допустимые связи; граф же отражает взаимодействия, фактически зарегистрированные в выбранном интервале. Не заменяет он и отдельную трассу: трасса описывает конкретное выполнение, граф — отношения, реконструированные по множеству таких выполнений.

Для проверки предложенного способа реконструкции разработан программный прототип [9], воспроизводящий описанный переход от трассировочных наблюдений к межсервисным отношениям и их характеристикам. Его задача состоит не в воспроизведении отдельной системы наблюдаемости, а в обеспечении перехода от трассировочных данных к множеству межсервисных отношений и рассчитанным для них характеристикам. Соответствие этапов реконструкции программным компонентам показано на рис. 1.

Программная реализация реконструкции межсервисных отношений

Исходные данные поступают в формате распределенных трасс, сформированных средствами OpenTelemetry [2] и переданных через OpenTelemetry Collector [10]. Контекст трассировки распространяется между сервисами в соответствии с W3C Trace Context [11], что позволяет связать операции, относящиеся к одному выполнению. На этапе обработки из трасс выделяются межсервисные вызовы, для них формируется идентифицирующий ключ, после чего однотипные наблюдения объединяются и используются для расчета характеристик соответствующих отношений.

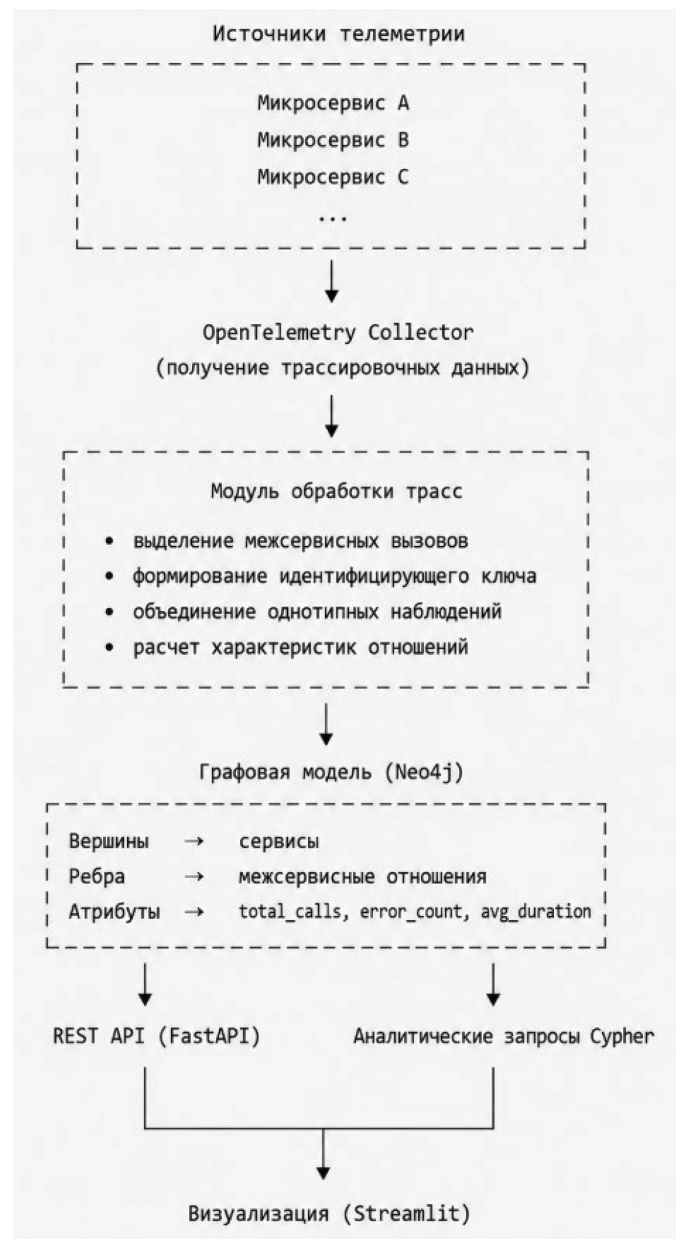


Рис. 1. Соответствие этапов реконструкции межсервисных отношений и компонентов программного прототипа

Трассировочные данные в данной работе рассматриваются как исходные для последующей реконструкции межсервисных отношений. Их получение требует инструментирования сервисов и передачи контекста трассировки в служебных заголовках, а сформированные span-данные создают дополнительный поток телеметрии к OpenTelemetry Collector [2, 10, 11].

Объем этого потока определяется интенсивностью трассируемых вызовов и составом

регистрируемых данных и не является постоянной величиной. Его количественная оценка в эксперимент не включалась, поскольку исследование начиналось с уже сформированного массива трассировочных наблюдений.

На этапе обработки из трасс выделяются межсервисные вызовы, для них формируется идентифицирующий ключ, после чего однотипные наблюдения объединяются и используются для расчета характеристик соответствующих отношений.

Результат сохраняется в Neo4j [12] в виде ориентированного атрибутированного графа, структура которого определена в предыдущем разделе. Такое представление обеспечивает соответствие между формализованной процедурой реконструкции и структурой данных, используемой при последующем экспериментальном анализе.

REST API, реализованный на FastAPI, и аналитические запросы Cypher обеспечивают доступ к полученной модели, а интерфейс на Streamlit — ее визуализацию. Эти компоненты выполняют вспомогательную функцию: они дают доступ к результату реконструкции, но не определяют сам способ его получения.

Экспериментальная проверка реализуемости реконструкции межсервисных отношений

Эксперимент воспроизводил на программном прототипе переход, формализованный ранее, — от отдельных межсервисных вызовов к характеристикам отношений, рассчитанным по множеству их реализаций. Проверялось, отражаются ли изменения исходных наблюдений в тех характеристиках реконструированного отношения, с которыми они связаны. Задача эксперимента этим и ограничивалась; эффективность диагностики, точность обнаружения аномалий и сравнение с другими средствами трассировки не оценивались.

Проверка проводилась на тестовой распределенной системе из пяти микросервисов: *api-gateway* обеспечивал прием и маршрутизацию запросов, *order-service* — обработку заказов, *inventory-service* — работу с данными о наличии, *payment-service* — обработку платежей, *notification-service* — формирование уведомлений. Для нее фор-

мировались синтетические трассировочные данные, воспроизводившие три состояния: штатную работу, повышенную задержку и ошибочные вызовы. Аналогичная конфигурация использовалась ранее при испытаниях прототипа [9]. Синтетические данные позволяли заранее задавать изменения отдельных вызовов и затем проследить, как они отражаются после реконструкции межсервисных отношений.

Все наборы трасс обрабатывались одинаково, различался только состав исходных наблюдений. Из зарегистрированных операций выделялись межсервисные вызовы, относящиеся к одной паре сервисов, после чего их значения накапливались на соответствующем ребре графа. В реализации прототипа обобщенный набор $A(R_k)$ представлен тремя показателями: *total_calls* — число вызовов, *error_count* — число ошибочных вызовов и *avg_duration* — средняя длительность. Первый соответствует n_k , отношение *error_count* к *total_calls* задает долю ошибок, а *avg_duration* представляет одну из статистических характеристик задержки.

Штатный сценарий использовался как базовый. В сценарии повышенной задержки увеличивалась длительность части вызовов, в сценарии ошибок часть вызовов завершалась с ошибочным статусом. Сопоставление этих изменений с характеристиками соответствующих межсервисных отношений приведено в таблице.

Изменения, заданные на уровне отдельных вызовов, получили отражение в характеристиках соответствующих межсервисных отношений. При увеличении длительности менялось значение *avg_duration*, а появление ошибочных завершений увеличивало *error_count* на том ребре, к которому были отнесены соответствующие вызовы.

На рис. 2 этот результат показан схематически для двух измененных сценариев: повышенной задержки и ошибочных вызовов. Рассчитанные значения относятся к конкретным направлениям взаимодействия между сервисами и закрепляются за соответствующими ребрами графовой модели.

Проверка проводилась на синтетических данных и небольшой тестовой системе, поэтому ее

Таблица

Проверка реконструкции межсервисных отношений по трассировочным наблюдениям

Экспериментальный сценарий	Заданное изменение наблюдений	Проверяемое соответствие	Результат в графовой модели
Штатная работа	Специальное изменение не вводится	Формирование межсервисных отношений и их базовых характеристик	Сформированы ребра с накопленными характеристиками
Повышенная задержка	Часть вызовов имеет увеличенную длительность	Отражение изменения длительности в характеристике соответствующего отношения	Изменение средней длительности отражено в атрибуте соответствующего ребра
Ошибочные вызовы	Часть вызовов завершается с ошибкой	Накопление ошибок в характеристиках соответствующего отношения	Число ошибочных вызовов отражено в атрибуте соответствующего ребра

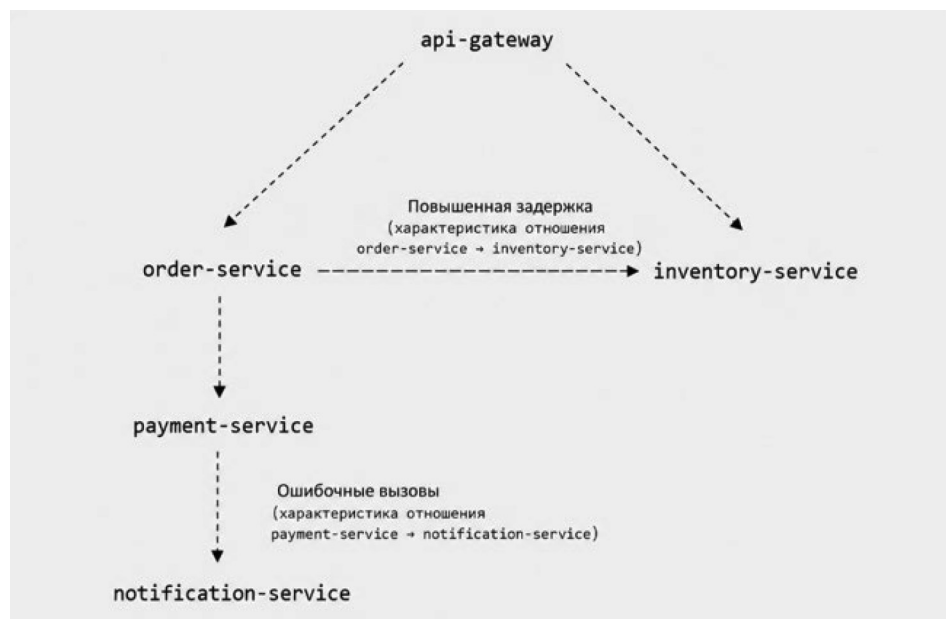


Рис. 2. Локализация характеристик реконструированных межсервисных отношений

результаты относятся только к контролируемым условиям эксперимента. Они подтверждают реализуемость предложенной процедуры, но не позволяют оценить устойчивость реконструируемых характеристик при изменчивой нагрузке, неполной трассировке или существенно большем числе взаимодействующих сервисов. Отдельной проверки требуют также влияние временного окна наблюдения и применение модели к производственным трассировочным данным.

Заключение

В работе формализован способ реконструкции межсервисных отношений по множеству трасси-

ровочных наблюдений. Однотипные вызовы объединяются по идентифицирующему ключу, после чего по их совокупности рассчитываются характеристики отношения. Полученные характеристики локализуются в ориентированной атрибутивной графовой модели относительно конкретных сервисов и направлений взаимодействия.

Программный прототип воспроизвел предложенную процедуру на синтетических трассировочных данных. В сценариях повышенной задержки и ошибочных вызовов изменения исходных наблюдений отразились в характеристиках соответствующих отношений, что подтвердило предусмотренное формализацией соответствие в контролируемых

условиях. Вопросы диагностической эффективности и точности обнаружения аномалий в эксперимент не входили.

Полученные результаты ограничены условиями проведенной проверки: использовались синтетические данные и небольшая тестовая система. Вопрос об устойчивости реконструируемых характеристик при изменчивой нагрузке, неполной трассировке

и большем числе взаимодействующих сервисов остается открытым; то же касается влияния временного окна наблюдения и перехода к производственным трассировочным данным. Отдельного исследования требует вопрос о том, при каком объеме и полноте наблюдений реконструируемые характеристики становятся достаточно устойчивыми для анализа поведения системы.

СПИСОК ИСТОЧНИКОВ

1. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure / B. H. Sigelman [et al.]. Google Technical Report. 2010. 14 p. URL: <http://research.google.com/pubs/archive/36356.pdf> (дата обращения: 05.08.2026).
2. Overview: OpenTelemetry Specification 1.60.0. URL: <http://opentelemetry.io/docs/specs/otel/overview> (дата обращения: 05.08.2026).
3. Leone J., Traini L. Enhancing Trace Visualizations for Microservices Performance Analysis // ICPE '23 Companion: Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (Coimbra, Portugal, 15–19 April 2023). New York: Association for Computing Machinery, 2023. Pp. 283–287. DOI: 10.1145/3578245.3584729
4. Automated Analysis of Distributed Tracing: Challenges and Research Directions / A. Bento [et al.] // Journal of Grid Computing. 2021. Vol. 19, no. 1. Art. no. 9. 15 p. DOI: 10.1007/s10723-021-09551-5
5. Visualizing Distributed Traces in Aggregate / A. Samanta [et al.] // ArXiv. 2024. Vol. 2412.07036. 10 p. DOI: 10.48550/arXiv.2412.07036
6. Aggregate-Driven Trace Visualizations for Performance Debugging / V. Anand [et al.] // ArXiv. 2020. Vol. 2010.13681. 7 p. DOI: 10.48550/arXiv.2010.13681
7. Graph: Kiali Documentation. URL: <http://kiali.io/docs/faq/graph> (дата обращения: 06.08.2026).
8. So, You Want to Trace Your Distributed System? Key Design Insights from Years of Practical Experience / R. R. Sambasivan [et al.]. Technical Report CMU-PDL-14-102. Parallel Data Laboratory of the Carnegie Mellon University. 2014. 25 p. URL: <http://www.pdl.cmu.edu/ftp/SelfStar/CMU-PDL-14-102.pdf> (дата обращения: 06.08.2026).
9. Дьяченко Н. И. Визуализация транзакций между микросервисами в реальном времени на основе графовых технологий: магистерская диссертация. СПб.: ПГУПС, 2026. 110 с.
10. Collector: OpenTelemetry Documentation. URL: <http://opentelemetry.io/docs/collector> (дата обращения: 05.08.2026).
11. Trace Context. W3C Recommendation. 23 November 2021. URL: <http://www.w3.org/TR/trace-context> (дата обращения: 05.08.2026).
12. Graph Database Concepts: Neo4j Documentation. URL: <http://neo4j.com/docs/getting-started/appendix/graphdb-concepts> (дата обращения: 05.08.2026).

Дата поступления: 24.08.2026

Решение о публикации: 29.08.2026

Reconstruction of Interservice Relationships from Tracing Data

- Andrey V. Zabrodin** — PhD in History, Associate Professor. Associate Professor of the “Information and Computing Systems” Department. Research interests: information systems, data analytics, programming paradigms, microservice architecture, cloud technologies. E-mail: zabrodin@pgups.ru
- Olga I. Khonina** — Assistant Lecturer of the “Information and Computing Systems” Department. Research interests: information systems, web development, programming paradigms. E-mail: olga.khonina.0311@mail.ru
- Kirill E. Fadkhutdinov** — 3rd-year Bachelor’s Degree Student in 09.03.01 Informatics and Computer Technology. Research interests: information systems, web development. E-mail: shiroasuri@gmail.com

Emperor Alexander I St. Petersburg State Transport University, 9 Moskovsky ave., Saint Petersburg, 190031, Russia

For citation: Zabrodin A. V., Khonina O. I., Fadkhutdinov K. E. Reconstruction of Interservice Relationships from Tracing Data, *Intellectual Technologies on Transport*, 2026, no. 3 (47), pp. 63–71. DOI: 10.20295/2413-2527-2026-347-63-71 (In Russian)

Abstract. *The problem of moving from the analysis of an individual distributed trace to the characteristics of interservice relationships is considered. Such characteristics are not available in a complete form within a single execution and can be identified only by comparing multiple tracing observations. **Objective:** to formalize a method for reconstructing interservice relationships from a set of tracing data and to verify its feasibility using a software prototype. **Methods:** extraction of interservice calls from distributed traces, grouping of similar observations by an identifying key, calculation of the characteristics of the corresponding relationships, and their representation in a directed attributed graph. **Results:** individual calls are represented as realizations of interservice interactions, while the characteristics of the relationships are determined from the aggregate set of such realizations. The proposed procedure was implemented in a software prototype and tested on synthetic tracing data from a five-component microservice system under normal operation, increased latency, and erroneous-call scenarios. The experiment demonstrated the reproducibility of the correspondence defined by the formalization between changes in individual observations and the characteristics of reconstructed interservice relationships. **Practical significance:** the resulting graph model makes it possible to associate characteristics that are not available in a complete form within a single trace with specific services and directions of interaction.*

Keywords: *distributed tracing, microservice architecture, interservice relationships, graph model, observability, OpenTelemetry, tracing data, system-level representation, interaction reconstruction*

REFERENCES

1. Sigelman B. H., et al. Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Google Technical Report, 2010, 14 p. Available at: <http://research.google.com/pubs/archive/36356.pdf> (accessed: August 05, 2026).
2. Overview: OpenTelemetry Specification 1.60.0. Available at: <http://opentelemetry.io/docs/specs/otel/overview> (accessed: August 05, 2026).
3. Leone J., Traini L. Enhancing Trace Visualizations for Microservices Performance Analysis, *ICPE ‘23 Companion: Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*, Coimbra, Portugal, April 15–19, 2023. New York, Association for Computing Machinery, 2023, pp. 283–287. DOI: 10.1145/3578245.3584729
4. Bento A., et al. Automated Analysis of Distributed Tracing: Challenges and Research Directions, *Journal of Grid Computing*, 2021, vol. 19, no. 1, art. no. 9, 15 p. DOI: 10.1007/s10723-021-09551-5

5. Samanta A., et al. Visualizing Distributed Traces in Aggregate, *ArXiv*, 2024, vol. 2412.07036, 10 p. DOI: 10.48550/arXiv.2412.07036
6. Anand V., et al. Aggregate-Driven Trace Visualizations for Performance Debugging, *ArXiv*, 2020, vol. 2010.13681, 7 p. DOI: 10.48550/arXiv.2010.13681
7. Graph: Kiali Documentation. Available at: <http://kiali.io/docs/faq/graph> (accessed: August 06, 2026).
8. Sambasivan R. R., et al. So, You Want to Trace Your Distributed System? Key Design Insights from Years of Practical Experience. Technical Report CMU-PDL-14-102. Parallel Data Laboratory of the Carnegie Mellon University, 2014, 25 p. Available at: <http://www.pdl.cmu.edu/ftp/SelfStar/CMU-PDL-14-102.pdf> (accessed: August 06, 2026).
9. Dyachenko N. I. Vizualizatsiya tranzaktsiy mezhdu mikroservisami v realnom vremeni na osnove grafovyykh tekhnologiy [Visualization of Transactions between Microservices in Real Time Based on Graph Technologies]: Master's Degree Thesis. Saint Petersburg, PSTU, 2026, 110 p. (In Russian)
10. Collector: OpenTelemetry Documentation. Available at: <http://opentelemetry.io/docs/collector> (accessed: August 05, 2026).
11. Trace Context. W3C Recommendation. 23 November 2021. Available at: <http://www.w3.org/TR/trace-context> (accessed: August 05, 2026).
12. Graph Database Concepts: Neo4j Documentation. Available at: <http://neo4j.com/docs/getting-started/appendix/graphdb-concepts> (accessed: August 05, 2026).

Received: August 24, 2026

Accepted: August 29, 2026