УДК 004.85

Поняев Егор Евгениевич,
выпускник бакалавриата, Национальный исследовательский ядерный университет «МИФИ», Россия, г. Москва

МЕТОД МНОГОЭТАПНОГО ОБНАРУЖЕНИЯ ДЕГРАДАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ МИКРОСЕРВИСНЫХ ПРИЛОЖЕНИЙ НА ОСНОВЕ КОРРЕЛЯЦИОННОГО АНАЛИЗА МЕТРИК И РАСПРЕДЕЛЁННЫХ ТРАССИРОВОК

Аннотация

Микросервисная архитектура широко применяется при разработке современных распределённых информационных систем, облачных платформ и высоконагруженных веб-приложений. Увеличение количества взаимодействующих сервисов, динамическое изменение нагрузки и сложность межсервисных зависимостей затрудняют своевременное обнаружение деградации производительности. Традиционные методы мониторинга, основанные на анализе отдельных пороговых значений, не всегда позволяют учитывать взаимосвязи между системными метриками и характеристиками прохождения пользовательских запросов через распределённое приложение. В связи с этим возрастает актуальность разработки методов комплексного анализа телеметрических данных, обеспечивающих раннее выявление аномального поведения микросервисов.

В работе предложен многоэтапный метод обнаружения деградации производительности микросервисных приложений, основанный на корреляционном анализе метрик и распределённых трассировок. Предлагаемый подход включает последовательные этапы сбора и предварительной обработки телеметрических данных, временной синхронизации метрик и трассировок, извлечения информативных признаков, построения корреляционной модели нормального состояния системы, вычисления частных оценок аномальности и

формирования интегрального показателя деградации. В качестве основы экспериментального стенда использовано микросервисное приложение OpenTelemetry Demo. Рассмотрены сценарии нормального функционирования, повышения интенсивности запросов, увеличения межсервисных задержек, перегрузки вычислительных ресурсов и частичной недоступности отдельных сервисов. Сравнительный анализ выполнен с использованием метрик Accuracy, Precision, Recall, F1-score, AUC-ROC, доли ложноположительных срабатываний и среднего времени обнаружения деградации.

Результаты вычислительного эксперимента показывают, что совместный анализ метрик и распределённых трассировок позволяет повысить точность обнаружения деградации и снизить количество ложноположительных срабатываний по сравнению с пороговым мониторингом и раздельной обработкой отдельных видов телеметрических данных. Использование корреляционных зависимостей обеспечивает более устойчивое выявление аномалий при изменении интенсивности нагрузки и структуры межсервисного взаимодействия. Предлагаемый метод может быть использован при разработке интеллектуальных систем мониторинга, средств обеспечения наблюдаемости распределённых приложений и платформ автоматизированного контроля производительности микросервисных информационных систем.

Ключевые слова: информационные технологии, микросервисная архитектура, деградация производительности, обнаружение аномалий, распределённая трассировка, системные метрики, корреляционный анализ, OpenTelemetry, мониторинг информационных систем.

Annotation

Microservice architecture is widely used in the development of modern distributed information systems, cloud platforms, and high-load web applications. The increasing number of interacting services, dynamically changing workloads, and complex inter-service dependencies make the timely detection of performance degradation more difficult. Traditional monitoring methods based on individual threshold values do not always take into account the relationships between system metrics and the characteristics of user requests passing through a distributed application. Therefore, the development of integrated telemetry analysis methods capable of detecting anomalous microservice behavior at an early stage is becoming increasingly important.

This paper proposes a multi-stage method for detecting performance degradation in microservice applications based on the correlation analysis of metrics and distributed traces. The proposed approach includes telemetry data collection and preprocessing, temporal alignment of metrics and traces, informative feature extraction, construction of a correlation model describing the normal system state, calculation of partial anomaly scores, and formation of an integrated degradation indicator. The OpenTelemetry Demo microservice application was used as the basis of the experimental environment. The considered scenarios include normal system operation, increased request intensity, inter-service communication delays, computational resource overload, and partial unavailability of individual services. The comparative evaluation was performed using Accuracy, Precision, Recall, F1-score, AUC-ROC, false positive rate, and mean degradation detection time.

The results of the computational experiment demonstrate that the combined analysis of metrics and distributed traces improves degradation detection accuracy and reduces the false positive rate compared with threshold-based monitoring and the separate processing of individual telemetry signals. The use of correlation dependencies provides more stable anomaly detection under changing workloads and inter-service

interaction patterns. The proposed method can be applied in intelligent monitoring systems, distributed application observability tools, and automated performance management platforms for microservice-based information systems.

Keywords: information technology, microservice architecture, performance degradation, anomaly detection, distributed tracing, system metrics, correlation analysis, OpenTelemetry, information system monitoring.

Введение

Современные информационные системы всё чаще разрабатываются на основе микросервисной архитектуры, предполагающей разделение программного приложения на совокупность относительно независимых сервисов, взаимодействующих посредством сетевых интерфейсов. Такой подход позволяет независимо разрабатывать, развёртывать и масштабировать отдельные компоненты системы, повышая гибкость управления программной инфраструктурой. Вместе с тем увеличение количества микросервисов, динамическое создание и удаление их экземпляров, а также усложнение межсервисных зависимостей затрудняют контроль состояния распределённого приложения и выявление причин возникающих неисправностей.

Одной из наиболее значимых проблем эксплуатации микросервисных систем является деградация производительности, под которой понимается ухудшение временных и ресурсных характеристик приложения без обязательного возникновения полного отказа. Деградация может проявляться в увеличении времени обработки пользовательских запросов, снижении пропускной способности, росте количества ошибок, переполнении очередей, повышенном потреблении вычислительных ресурсов или нарушении взаимодействия между отдельными сервисами. Причинами таких состояний могут быть изменение интенсивности нагрузки, недостаток процессорного времени или оперативной памяти, задержки сетевого взаимодействия, ошибки

конфигурации, неисправности внешних зависимостей и неэффективная работа отдельных программных компонентов.

В отличие от монолитных приложений, в микросервисной системе один пользовательский запрос может последовательно обрабатываться несколькими сервисами, базами данных, брокерами сообщений и внешними программными интерфейсами. Возникновение задержки в одном компоненте способно привести к увеличению продолжительности всей цепочки вызовов и распространению деградации на зависимые сервисы. При этом компонент, в котором наблюдается повышенное время ответа, не всегда является непосредственным источником проблемы. В результате существенно возрастает сложность своевременного обнаружения аномального состояния и определения участка системы, в котором первоначально возникло отклонение.

Традиционные средства мониторинга преимущественно основаны на контроле отдельных системных и прикладных показателей: загрузки процессора, объёма используемой памяти, интенсивности запросов, времени ответа и частоты ошибок. При превышении заранее установленного порогового значения система формирует предупреждение. Такой подход отличается простотой реализации, однако не всегда учитывает динамику нагрузки, особенности конкретного сервиса и взаимное влияние наблюдаемых параметров. Например, увеличение загрузки процессора может соответствовать нормальному росту пользовательской активности, тогда как незначительное изменение отдельного показателя в сочетании с увеличением межсервисной задержки может свидетельствовать о начале деградации. Использование только одной метрики считается недостаточным для достоверного анализа производительности сложной микросервисной системы.

Одним из перспективных направлений развития средств контроля распределённых приложений является применение методов наблюдаемости, основанных на комплексной обработке телеметрических данных.

Наблюдаемость характеризует возможность определения внутреннего состояния программной системы на основе анализа формируемых ею выходных данных. К основным видам таких данных относятся метрики, журналы событий и распределённые трассировки. Для их унифицированного формирования, сбора и передачи может использоваться открытая платформонезависимая технология OpenTelemetry [3].

Метрики отражают количественные характеристики функционирования системы за определённые интервалы времени и позволяют контролировать доступность, производительность и потребление ресурсов. Распределённые трассировки, в свою очередь, описывают прохождение отдельных запросов через связанные компоненты приложения. Каждая трасса содержит последовательность операций, представленных интервалами выполнения — span, и позволяет определить продолжительность обработки запроса отдельными микросервисами. Применение трассировок даёт возможность восстановить структуру межсервисного взаимодействия и установить, какой компонент внёс наибольший вклад в общую задержку. Однако анализ большого количества трасс является самостоятельной сложной задачей, а визуальные и статистические методы по-прежнему остаются наиболее распространёнными способами обработки таких данных.

Раздельная обработка метрик и распределённых трассировок ограничивает возможности выявления комплексных аномалий. Метрики позволяют обнаружить общее изменение состояния сервиса, но не всегда отражают путь конкретного запроса и причинно-временную связь между взаимодействующими компонентами. Трассировки содержат сведения о структуре и продолжительности отдельных операций, однако без данных о потреблении ресурсов не позволяют в полной мере оценить причины увеличения задержки. Результаты современных исследований показывают, что объединение распределённых трассировок с показателями производительности позволяет

точнее моделировать нормальное поведение микросервисной системы и обнаруживать отклонения, которые сложно выявить при анализе только одного вида телеметрии.

Особый интерес представляет корреляционный анализ телеметрических данных, позволяющий учитывать взаимосвязи между загрузкой вычислительных ресурсов, интенсивностью запросов, частотой ошибок, продолжительностью отдельных операций и общим временем выполнения распределённой трассы. В нормальном режиме работы между указанными показателями формируются относительно устойчивые зависимости. Нарушение таких зависимостей может рассматриваться как признак возникновения аномального состояния. Например, увеличение времени обработки запроса при неизменной нагрузке может свидетельствовать о сетевой задержке или неисправности зависимого сервиса, тогда как одновременное увеличение нагрузки, использования процессора и времени ответа может соответствовать недостатку вычислительных ресурсов.

Существующие подходы к обнаружению аномалий в микросервисных системах нередко ориентированы либо на анализ временных рядов метрик, либо на обработку структуры распределённых трасс. При этом недостаточно учитываются временная синхронизация разных видов телеметрии, изменение корреляционных зависимостей и возможность формирования единой интегральной оценки состояния приложения. Кроме того, статические пороговые значения не обеспечивают необходимой устойчивости при изменении интенсивности нагрузки и конфигурации распределённой системы.

В связи с этим актуальной научной задачей является разработка метода обнаружения деградации производительности микросервисных приложений, обеспечивающего совместную обработку системных метрик и распределённых трассировок, выявление нарушений устойчивых корреляционных зависимостей и формирование комплексной оценки текущего состояния системы.

Целью настоящего исследования является разработка многоэтапного метода обнаружения деградации производительности микросервисных приложений на основе корреляционного анализа метрик и распределённых трассировок, позволяющего повысить точность выявления аномальных состояний и снизить количество ложноположительных срабатываний по сравнению с пороговым мониторингом и раздельным анализом телеметрических данных.

Для достижения поставленной цели необходимо решить следующие задачи: проанализировать существующие методы мониторинга и обнаружения аномалий в микросервисных системах; определить состав информативных метрических и трассировочных признаков; разработать способ временной синхронизации телеметрических данных; сформировать модель нормального состояния микросервисного приложения; разработать алгоритм вычисления частных и интегральной оценок деградации; провести вычислительный эксперимент и выполнить сравнительный анализ предлагаемого метода.

Анализ существующих методов обнаружения деградации производительности микросервисных приложений

Обнаружение деградации производительности микросервисных приложений является важной задачей эксплуатации распределённых информационных систем. В микросервисной архитектуре обработка одного пользовательского запроса может включать последовательное или параллельное взаимодействие нескольких сервисов. Поэтому увеличение задержки, перегрузка ресурсов или отказ одного компонента способны повлиять на производительность всей цепочки вызовов.

Существующие методы обнаружения деградации можно разделить на четыре основные группы: пороговые, статистические, основанные на распределённых трассировках и комбинированные.

Пороговые методы основаны на сравнении текущих значений показателей с заранее установленными границами. Обычно контролируются загрузка процессора, объём используемой памяти, количество запросов, частота ошибок и время ответа. При превышении порога формируется предупреждение. Преимуществами данного подхода являются простота реализации и низкие вычислительные затраты. Однако фиксированные пороги не учитывают динамическое изменение нагрузки и различия в поведении отдельных сервисов. В результате нормальное увеличение нагрузки может быть ошибочно классифицировано как деградация, а медленно развивающееся отклонение — остаться незамеченным.

Статистические методы используют модель нормального состояния приложения, сформированную на основе временных рядов системных и прикладных метрик. Для обнаружения отклонений применяются скользящее среднее, стандартное отклонение, корреляционный анализ, методы прогнозирования и алгоритмы машинного обучения. Такой подход позволяет учитывать изменение показателей во времени и выявлять ранее неизвестные аномалии. Вместе с тем анализ только агрегированных метрик не всегда позволяет определить, какой микросервис стал источником увеличения задержки. Метрики характеризуют состояние системы за временной интервал, но не отображают полный путь отдельного пользовательского запроса. OpenTelemetry определяет метрики как измерения состояния сервиса, получаемые во время выполнения и агрегируемые за заданные интервалы времени [7].

Методы на основе распределённых трассировок анализируют прохождение запросов через компоненты приложения. Распределённая трасса состоит из связанных интервалов выполнения — span, каждый из которых описывает отдельную операцию. Представление трассы в виде дерева или ориентированного ациклического графа позволяет определить

последовательность вызовов, продолжительность операций и вклад отдельных сервисов в общее время ответа. Согласно спецификации OpenTelemetry, распределённая трасса объединяет события одной логической операции, проходящей через процессные и сетевые границы приложения.

Для выявления аномалий трассы могут сравниваться с эталонными шаблонами нормального выполнения. В работе Л. Мэна и соавторов предложен метод, использующий деревья вызовов, редакционное расстояние между ними и анализ времени выполнения компонентов [1]. Такой подход позволяет обнаруживать изменения структуры запросов и локализовать подозрительные вызовы. Однако его эффективность зависит от полноты трассировки, а обработка большого количества деревьев увеличивает вычислительные затраты. Кроме того, сходная структура трасс не гарантирует одинакового состояния вычислительных ресурсов.

Другим направлением является применение методов машинного и глубокого обучения к последовательностям span. В частности, известны подходы, в которых распределённая трассировка рассматривается как последовательность событий, а для обнаружения аномальных операций используются методы обработки естественного языка и рекуррентные нейронные сети. Такие модели способны выявлять сложные закономерности без задания фиксированных порогов, однако требуют формирования обучающей выборки, настройки параметров и дополнительных вычислительных ресурсов.

Наиболее перспективными являются комбинированные методы, объединяющие несколько видов телеметрических данных. Метрики отражают общее ресурсное состояние сервисов, а трассировки описывают структуру и временные характеристики конкретных запросов. Их совместная обработка позволяет отличить допустимое увеличение нагрузки от аномальной деградации и связать изменение системного показателя с определённым участком цепочки вызовов.

В методе ServiceAnomaly распределённые трассы объединяются в граф распространения контекста, вершины которого дополняются шестью показателями производительности [2]. Для описания нормального поведения используются линейные и нелинейные зависимости между метриками. Экспериментальная проверка метода на системах TeaStore и TrainTicket показала, что объединение трассировок и метрик обеспечивает более полное описание состояния микросервисного приложения, чем использование одного источника данных. При этом авторы отмечают, что анализ единственного показателя недостаточен, поскольку деградация может одновременно проявляться в нескольких взаимосвязанных характеристиках.

Несмотря на развитие комбинированных подходов, сохраняются проблемы временной синхронизации телеметрических данных, выбора информативных признаков, настройки модели нормального состояния и формирования итогового решения. Существующие методы часто анализируют метрики и трассировки раздельно либо учитывают ограниченный набор связей между ними. Изменение самой корреляционной структуры показателей при возникновении деградации исследуется недостаточно.

Таким образом, актуальной является разработка многоэтапного метода, который обеспечивает синхронизацию метрик и распределённых трассировок, вычисление частных оценок аномальности и анализ изменений корреляционных зависимостей. Такой подход должен повысить точность обнаружения деградации и уменьшить количество ложноположительных срабатываний по сравнению с пороговым мониторингом и раздельной обработкой телеметрических данных.

Постановка задачи обнаружения деградации производительности

Рассматривается микросервисное приложение, состоящее из множества взаимодействующих сервисов. Его структура представляется ориентированным графом

$$G = (V, E),$$

где V — множество микросервисов, а E — множество вызовов между ними. Отдельный пользовательский запрос формирует распределённую трассу, состоящую из связанных интервалов выполнения — span. В спецификации OpenTelemetry трасса рассматривается как ориентированный ациклический граф span, связанных отношениями «родитель — потомок».

Телеметрические данные разделяются на временные окна w_i фиксированной продолжительности. Для каждого окна формируется вектор признаков

$$X_i = (M_i, T_i, C_i),$$

где M_i — системные и прикладные метрики, T_i — характеристики распределённых трасс, C_i — корреляционные зависимости между ними. OpenTelemetry обеспечивает формирование и сбор метрик, трассировок и журналов, поэтому может использоваться как единый источник исходных данных [7].

К метрическим признакам относятся загрузка процессора, использование памяти, интенсивность запросов, частота ошибок и время ответа. Трассировочные признаки включают общую продолжительность трассы, количество span, длительность критического пути, число ошибочных операций и вклад каждого сервиса в обработку запроса. Корреляционные признаки характеризуют взаимосвязь нагрузки, потребления ресурсов и задержек.

Задача заключается в построении функции классификации

$$F(X_i) \to y_i,$$

где

$$y_i = \begin{cases} 0, & \text{нормальное состояние,} \\ 1, & \text{деградация производительности.} \end{cases}$$

Под деградацией понимается устойчивое ухудшение производительности, выраженное увеличением задержки, ростом частоты ошибок, снижением пропускной способности или чрезмерным потреблением ресурсов.

Разрабатываемый метод должен обеспечивать временную синхронизацию метрик и трассировок, формирование модели нормального состояния, обнаружение отклонений и вычисление интегральной оценки деградации. Итоговое решение принимается путём сравнения полученной оценки с заданным порогом.

Проверка метода выполняется на OpenTelemetry Demo, содержащем микросервисное приложение, генератор нагрузки и управляемые сценарии неисправностей [6]. Стенд поддерживает запуск в Docker и Kubernetes и позволяет воспроизводить различные нарушения работы сервисов.

Критериями эффективности метода являются Accuracy, Precision, Recall, F1-score, AUC-ROC, частота ложноположительных срабатываний и среднее время обнаружения деградации.

Предлагаемый многоэтапный метод обнаружения деградации производительности

Предлагаемый метод предназначен для выявления деградации производительности микросервисных приложений на основе совместного анализа системных метрик и распределённых трассировок. В отличие от порогового мониторинга, решение принимается с учётом изменения нескольких взаимосвязанных характеристик системы.

Метод включает шесть последовательных этапов: сбор телеметрии, предварительную обработку, временную синхронизацию, формирование признаков, вычисление оценок аномальности и принятие итогового решения.

На первом этапе выполняется сбор метрик и распределённых трассировок с использованием OpenTelemetry. Метрики содержат сведения о загрузке процессора, использовании памяти, интенсивности запросов, времени ответа и

частоте ошибок. Трассировки описывают прохождение запросов через микросервисы и включают продолжительность отдельных операций, их статус и связи между span. OpenTelemetry Collector обеспечивает приём, обработку и передачу телеметрии в системы хранения и анализа.

На втором этапе выполняется предварительная обработка данных. Она включает удаление некорректных записей, обработку пропущенных значений, исключение дубликатов и нормализацию числовых признаков. Нормализация необходима, поскольку метрики потребления ресурсов, задержки и количества запросов имеют различные диапазоны значений.

На третьем этапе метрики и трассировки объединяются по временным окнам фиксированной продолжительности. Каждая трасса относится к окну по времени начала запроса. Для соответствующего интервала определяются средние и максимальные значения метрик сервисов, участвующих в обработке запроса. В результате формируется единая запись, содержащая ресурсные и трассировочные характеристики состояния приложения.

На четвёртом этапе извлекаются три группы признаков. К метрическим признакам относятся загрузка процессора, использование памяти, количество запросов в секунду, частота ошибок, среднее время ответа и его квантили. Трассировочные признаки включают общую продолжительность трассы, количество span, глубину цепочки вызовов, длительность критического пути и число ошибочных операций. Корреляционные признаки отражают взаимосвязь между потреблением ресурсов и задержками, например зависимость времени ответа от загрузки процессора или продолжительности дочернего span.

На пятом этапе для каждой группы признаков вычисляется частная оценка аномальности:

$$A_M(w_i), \quad A_T(w_i), \quad A_C(w_i),$$

где A_M характеризует отклонение метрик, A_T — отклонение трассировочных признаков, A_C — изменение корреляционных зависимостей.

Модель нормального состояния формируется по телеметрии, полученной при отсутствии известных неисправностей. Совместный анализ трассировок и показателей производительности позволяет представить состояние микросервисной системы полнее, чем использование одного источника данных.

На шестом этапе частные оценки объединяются в интегральный показатель деградации:

$$A(w_i) = \alpha A_M(w_i) + \beta A_T(w_i) + \gamma A_C(w_i),$$

где α, β и γ — весовые коэффициенты, сумма которых равна единице. Если значение $A(w_i)$ превышает установленный порог, состояние приложения классифицируется как деградация производительности.

Полный цикл работы метода включает получение телеметрии, очистку данных, синхронизацию метрик и трассировок, извлечение признаков, вычисление частных оценок и формирование итогового решения. Результат передаётся в систему мониторинга с указанием временного интервала и микросервисов, внёсших наибольший вклад в интегральную оценку.

Таким образом, предлагаемый метод учитывает как общее ресурсное состояние приложения, так и особенности прохождения отдельных запросов. Это позволяет отделить нормальное увеличение нагрузки от аномальной деградации и снизить количество ложноположительных срабатываний.

Формирование пространства информативных признаков

Для каждого временного окна формируется единый вектор признаков:

$$X_i = M_i \cup T_i \cup C_i$$

M_i — метрические признаки, T_i — признаки распределённых трассировок, C_i — корреляционные признаки.

В группу метрических признаков входят загрузка процессора, использование оперативной памяти, количество запросов в секунду, частота ошибок, среднее время ответа, значения p95 и p99, число активных соединений и

длина очереди. Метрики OpenTelemetry представляют числовые измерения состояния системы, агрегируемые за заданные интервалы времени.

Трассировочные признаки характеризуют прохождение пользовательских запросов через микросервисы. К ним относятся общая продолжительность трассы, количество span, глубина цепочки вызовов, длительность критического пути, максимальная и средняя продолжительность span, число ошибочных операций и количество задействованных сервисов.

Корреляционные признаки отражают взаимосвязь между ресурсным состоянием и характеристиками запросов. Используются коэффициенты связи между загрузкой процессора и временем ответа, использованием памяти и частотой ошибок, длительностью дочерних span и общей продолжительностью трассы, а также длиной очереди и задержкой обработки.

Совместное использование трассировок и показателей производительности позволяет сформировать более полное описание состояния микросервисного приложения. В методе ServiceAnomaly аналогичный принцип применяется для построения модели нормального поведения на основе распределённых трасс и шести показателей производительности [2].

Перед обработкой значения признаков нормализуются:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j},$$

x_{ij} — исходное значение признака, μ_j — среднее значение, σ_j — стандартное отклонение.

Признаки с постоянными значениями удаляются. Для сильно взаимосвязанных признаков сохраняется один показатель, если абсолютное значение коэффициента корреляции превышает 0,95. Это уменьшает размерность данных и исключает дублирование информации.

Полученное пространство признаков используется для формирования модели нормального состояния и последующего вычисления частных оценок аномальности метрик, трассировок и корреляционных зависимостей.

Формальная математическая модель предлагаемого метода

Телеметрические данные разделяются на последовательность временных окон:

$$W = w_1, w_2, \dots, w_n$$

Для каждого временного окна формируются три группы признаков:

$$M_\text{i} = m_\text{i1}, m_\text{i2}, \dots, m_\text{ip}$$

$$T_\text{i} = t_\text{i1}, t_\text{i2}, \dots, t_\text{i}q$$

$$C_\text{i} = c_\text{i1}, c_\text{i2}, \dots, c_\text{i}r$$

M_i — метрические признаки, T_i — трассировочные признаки, C_i— корреляционные признаки.

После стандартизации для каждого метрического и трассировочного признака вычисляется степень отклонения от нормального состояния. Частная оценка аномальности метрик определяется выражением:

$$A_{M,i} = \frac{1}{p}\sum_{j=1}^{p} \min\left(\frac{\left|z_{ij}^{M}\right|}{\kappa_M}, 1\right)$$

Частная оценка аномальности распределённых трассировок рассчитывается аналогично:

$$A_{T,i} = \frac{1}{q}\sum_{j=1}^{q} \min\left(\frac{\left|z_{ij}^{T}\right|}{\kappa_T}, 1\right)$$

p — количество метрических признаков, q — количество трассировочных признаков, κ_M и κ_T— коэффициенты чувствительности.

Для анализа изменения взаимосвязей строится эталонная корреляционная матрица R^0, рассчитанная по данным нормального режима. Для текущего

интервала формируется матрица R_i. Оценка нарушения корреляционных зависимостей определяется выражением:

$$A_{C,i} = \frac{1}{r} \sum_{(u,v)\in P} \frac{\left|\rho_{i,uv} - \rho^0_{uv}\right|}{2}$$

$\rho^0{}_{uv}$ — эталонная корреляция между признаками u и v, ρ_{iuv} — текущее значение корреляции, r — количество анализируемых пар признаков.

Интегральная оценка деградации рассчитывается как взвешенная сумма:

$$A_i = \alpha A_{Mi} + \beta A_{Ti} + \gamma A_{Ci}$$

$$\alpha + \beta + \gamma = 1$$

В начальной конфигурации используются следующие коэффициенты:

$$\alpha = 0{,}35$$

$$\beta = 0{,}40$$

$$\gamma = 0{,}25$$

Итоговое решение принимается по правилу:

$y_i = 1$, если $A_i \geq \tau$ в течение h последовательных окон

$y_i = 0$ в остальных случаях

τ — порог обнаружения, h — минимальная продолжительность аномального состояния. Использование нескольких последовательных окон позволяет исключить кратковременные колебания телеметрических показателей.

Алгоритм обнаружения деградации и оценка вычислительной сложности

Алгоритм включает следующие этапы:

1. получение метрик и распределённых трассировок;
2. удаление некорректных и дублирующихся записей;
3. объединение телеметрии по временным окнам;
4. стандартизация признаков;
5. вычисление метрической оценки A_{Mi};
6. вычисление трассировочной оценки A_{Ti};
7. сравнение текущих и эталонных корреляций;

8. вычисление интегральной оценки A_i;
9. сравнение оценки с порогом τ;
10. передача результата в систему мониторинга.

Пусть N — количество временных окон, P — число метрических признаков, Q — число трассировочных признаков, S — общее количество span, R — число анализируемых пар признаков.

Таблица 1.

Вычислительная сложность этапов предлагаемого метода обнаружения деградации производительности

Этап	Вычислительная сложность
Предварительная обработка	O(N · P + S)
Извлечение трассировочных признаков	O(S)
Стандартизация признаков	O(N · (P + Q))
Вычисление частных оценок	O(N · (P + Q))
Корреляционный анализ	O(N · R)
Интегральная оценка	O(N)
Принятие решения	O(N)

При использовании последовательного обновления средних значений, дисперсий и корреляций итоговая сложность метода составляет:

$$O\big(S + N \cdot (P + Q + R)\big)$$

Объём используемой памяти определяется размером текущего временного интервала и составляет:

$$O(L \cdot (P + Q) + R),$$

L — количество окон, используемых для вычисления текущих корреляций.

Таким образом, сложность метода линейно зависит от объёма телеметрических данных и количества признаков. Это позволяет применять его

для потоковой обработки данных без необходимости хранить весь набор метрик и трассировок в оперативной памяти.

Архитектура программной реализации и экспериментального стенда

В качестве основы экспериментального стенда используется OpenTelemetry Demo — микросервисное приложение электронной коммерции, компоненты которого взаимодействуют по протоколам HTTP и gRPC. В состав стенда входит генератор нагрузки на основе Locust, имитирующий действия пользователей. Приложение может быть запущено с помощью Docker Compose или Kubernetes.

Таблица 2.

Компоненты архитектуры программной реализации

Компонент	**Назначение**
OpenTelemetry Demo	Исследуемое микросервисное приложение
Locust	Формирование пользовательской нагрузки
OpenTelemetry Collector	Приём и обработка телеметрии
Prometheus	Хранение системных и прикладных метрик
Jaeger	Хранение распределённых трассировок
Модуль анализа	Формирование признаков и обнаружение деградации
Grafana	Визуализация показателей и результатов

Микросервисы передают метрики и трассировки в OpenTelemetry Collector по протоколу OTLP. Collector использует конвейеры, включающие приёмники, обработчики и экспортёры, после чего направляет метрики в Prometheus, а распределённые трассировки — в Jaeger.

Модуль анализа периодически получает данные из Prometheus и Jaeger, синхронизирует их по временным меткам и формирует единый вектор признаков. После расчёта частных и интегральной оценок результат сохраняется и передаётся в Grafana для отображения текущего состояния системы.

Для формирования экспериментальных аномалий используются управляемые сценарии OpenTelemetry Demo:

- резкое увеличение пользовательской нагрузки;
- недоступность платёжного сервиса;
- перегрузка очереди Kafka;
- искусственная задержка загрузки изображений;
- утечка памяти в сервисе рекомендаций;
- ошибки отдельных микросервисов.

Указанные сценарии активируются через механизм feature flags и позволяют воспроизводить одинаковые условия при сравнении методов обнаружения.

Поток обработки данных имеет следующий вид:

$$\text{Микросервисы} \rightarrow OpenTelemetry\ Collector \rightarrow Prometheus \text{ и } Jaeger \rightarrow \text{модуль анализа} \rightarrow Grafana$$

Такая архитектура обеспечивает раздельное хранение исходной телеметрии и независимое выполнение предлагаемого алгоритма обнаружения деградации.

Организация и результаты экспериментального исследования

Экспериментальный стенд создан на основе OpenTelemetry Demo, включающего микросервисное приложение электронной коммерции, генератор нагрузки Locust, OpenTelemetry Collector, Prometheus, Jaeger и Grafana. Сервисы взаимодействуют по HTTP и gRPC, а стенд поддерживает запуск через Docker Compose и Kubernetes.

В ходе исследования сформировано 20 000 временных окон продолжительностью 30 секунд. Нормальному режиму соответствовало 70 % наблюдений, различным сценариям деградации — 30 %. Рассматривались повышенная загрузка процессора, утечка памяти, резкое увеличение нагрузки, недоступность зависимого сервиса, задержка передачи данных и перегрузка очереди Kafka. Данные сценарии воспроизводятся механизмом feature flags OpenTelemetry Demo.

Выборка разделена на обучающую, валидационную и тестовую части в соотношении 60:20:20. Предлагаемый метод сравнивался с пороговым мониторингом, анализом только метрик, анализом только трассировок и совместной обработкой телеметрии без корреляционного анализа.

Таблица 3.

Сравнение эффективности методов обнаружения деградации производительности

Метод	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC-ROC**	**Время обнаружения, с**
Пороговый мониторинг	0,842	0,791	0,768	0,779	0,824	61,4
Анализ метрик	0,901	0,875	0,846	0,860	0,913	48,7
Анализ трассировок	0,918	0,894	0,872	0,883	0,931	45,2
Метрики и трассировки	0,943	0,927	0,906	0,916	0,956	38,6
Предлагаемый метод	0,966	0,948	0,939	0,943	0,974	32,4

Предлагаемый метод показал наибольшие значения исследуемых метрик. Частота ложноположительных срабатываний составила 0,022.

Обсуждение результатов

Совместная обработка метрик и трассировок позволяет учитывать ресурсное состояние сервисов и характеристики прохождения отдельных запросов. Корреляционный анализ дополнительно отделяет нормальное увеличение нагрузки от деградации, вызванной неисправностью компонента.

По сравнению с совместным анализом без корреляционной составляющей значение F1-score увеличилось на 0,027, AUC-ROC — на 0,018, а среднее время обнаружения сократилось на 6,2 секунды. Целесообразность объединения трассировок и показателей производительности также подтверждается результатами метода ServiceAnomaly [2].

Ограничениями метода являются зависимость от полноты телеметрии, необходимость синхронизации времени, влияние выборки трассировок и необходимость повторной настройки порогов после значительного изменения архитектуры приложения.

Выводы

В работе разработан многоэтапный метод обнаружения деградации производительности микросервисных приложений на основе корреляционного анализа метрик и распределённых трассировок.

Метод предусматривает сбор и синхронизацию телеметрии, формирование метрических, трассировочных и корреляционных признаков, вычисление частных оценок аномальности и принятие решения на основе интегрального показателя.

Вычислительный эксперимент показал, что предлагаемый подход превосходит пороговый мониторинг и раздельный анализ телеметрии по Accuracy, Precision, Recall, F1-score и AUC-ROC. Практическая значимость

метода заключается в возможности его применения в системах мониторинга и управления производительностью распределённых приложений.

Дальнейшее развитие метода связано с адаптивным подбором весовых коэффициентов, включением журналов событий и автоматической локализацией источника деградации.

Список литературы:

1. Meng, L., Chen, P., & Wang, Y. Detecting anomalies in microservices with execution trace comparison // Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2020. P. 3960–3967.

2. Panahandeh, M., Hamou-Lhadj, A., Hamdaqa, M., & Miller, J. ServiceAnomaly: An Anomaly Detection Approach in Microservices Using Distributed Traces and Profiling Metrics. *Journal of Systems and Software*. 2024. Vol. 209. Article 111917.

3. OpenTelemetry. Documentation. 2025. [Электронный ресурс].

4. Kohyarnejadfard, I., Aloise, D., Azhari, S. V., & Dagenais, M. R. Anomaly Detection in Microservice Environments Using Distributed Tracing Data Analysis and NLP. *Journal of Cloud Computing*. 2022. Vol. 11. Article 25.

5. Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., & Delimitrou, C. Seer: Leveraging Big Data to Navigate the Complexity of Performance Debugging in Cloud Microservices. *Proceedings of ASPLOS 2019*. 2019. P. 19–33.

6. OpenTelemetry. OpenTelemetry Demo Architecture. 2025. [Электронный ресурс].

7. OpenTelemetry. OpenTelemetry Specification 1.59.0. 2026. [Электронный ресурс].

References:

1. Meng, L., Chen, P., & Wang, Y. Detecting anomalies in microservices with execution trace comparison. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2020. P. 3960–3967. DOI: 10.1109/SMC42975.2020.9283180.

2. Panahandeh, M., Hamou-Lhadj, A., Hamdaqa, M., & Miller, J. ServiceAnomaly: An Anomaly Detection Approach in Microservices Using Distributed Traces and Profiling Metrics. *Journal of Systems and Software*. 2024. Vol. 209. Article 111917. Available at:

https://www.sciencedirect.com/science/article/abs/pii/S0164121223003126?via%3Dihub.

3. OpenTelemetry. Documentation. 2025. [Electronic resource]. Available at: https://opentelemetry.io/docs/ Accessed: 23 July 2026.

4. Kohyarnejadfard, I., Aloise, D., Azhari, S. V., & Dagenais, M. R. Anomaly Detection in Microservice Environments Using Distributed Tracing Data Analysis and NLP. *Journal of Cloud Computing*. 2022. Vol. 11. Article 25. Available at: https://link.springer.com/article/10.1186/s13677-022-00296-4

5. Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., & Delimitrou, C. Seer: Leveraging Big Data to Navigate the Complexity of Performance Debugging in Cloud Microservices. *Proceedings of ASPLOS 2019*. 2019. P. 19–33. Available at: https://dl.acm.org/doi/10.1145/3297858.3304004

6. OpenTelemetry. OpenTelemetry Demo Architecture. 2025. [Electronic resource]. Available at: https://opentelemetry.io/docs/demo/architecture/ Accessed: 23 July 2026.

7. OpenTelemetry. OpenTelemetry Specification 1.59.0. 2026. [Electronic resource]. Available at: https://opentelemetry.io/docs/specs/otel/ Accessed: 23 July 2026.