УДК [004.738.5:339.138]-047.36
DOI: 10.21122/2309-4923-2025-3-29-33

ПИСКУН Е. С., КОТЬКО Е. Н.

АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ И УСТОЙЧИВОСТИ СИСТЕМЫ МОНИТОРИНГА ПЛАТФОРМЫ ЭЛЕКТРОННОЙ КОММЕРЦИИ НА ОСНОВЕ PROMETHEUS И GRAFANA

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Аннотация. *Разработан Python-сервис мониторинга пользовательских метрик, взаимодействующий с открытым API маркетплейса. Архитектура реализована с использованием Prometheus и Grafana и ориентирована на контроль производительности ключевых этапов обработки данных: количества запросов, ошибок, времени отклика, скорости записи в базу данных и характеристик товаров. Для оценки устойчивости выполнено моделирование аварийных ситуаций, включая сбои внешних API, деградацию базы данных, сетевые задержки, рост нагрузки и утечки памяти. Применение потоковой обработки данных в сочетании с SQLite обеспечивает высокую производительность и надёжность.*

Ключевые слова: *мониторинг, распределённые системы, Python-сервис, маркетплейс, Prometheus, Grafana, многопоточность, производительность, телеметрия, стресс-тестирование, обработка данных, API-интеграция*

Введение

Современные информационные системы характеризуются высокой сложностью и распределённой архитектурой, что требует постоянного мониторинга ключевых компонентов, особенно при работе с внешними API и системами хранения данных [1]. Для маркетплейсов критично своевременное обновление информации о товарах, заказах и остатках, так как сбои приводят к потерям и снижению лояльности пользователей. Мониторинг с использованием Prometheus и Grafana позволяет выявлять отклонения, фиксировать ошибки и анализировать их причины [2; 3]. В работе реализован Python-сервис мониторинга пользовательских метрик при обработке данных с открытого API маркетплейса, использующий многопоточность (ThreadPoolExecutor) и SQLite. Регистрируются ключевые параметры производительности, данные экспортируются в Prometheus и визуализируются в Grafana, обеспечивая прототип мониторинга для маркетплейсов и e-commerce-платформ [3; 4].

Архитектура и реализации системы мониторинга

Объект исследования – Python-сервис для автоматизированного извлечения, хранения и анализа данных из открытого API маркетплейса на примере платформы электронной коммерции среднего масштаба, обеспечивающий регистрацию ошибок, измерение производительности и визуализацию ключевых показателей.

Архитектура системы описана с использованием C4-нотации, что формализует решения на разных уровнях – от контекста до модулей кода [5].

Контекстная диаграмма системы мониторинга выступает связующим звеном между пользователем и внешними сервисами. Python-сервис извлекает данные из API маркетплейса и передает телеметрию в Prometheus, где она используется для визуализации в Grafana.

На уровне контейнеров (в терминах C4-нотации) архитектура системы включает несколько основных компонентов. Python-сервис обрабатывает данные и формирует метрики, SQLite сохраняет товарные данные, Prometheus собирает и хранит метрики, Grafana визуализирует их. Сервис регулярно опрашивает API, заносит результаты в базу данных и публикует метрики, которые автоматически собираются Prometheus. Под контейнером в работе понимается исполняемое окружение (например, сервис или база данных).

Компонентная диаграмма (рисунок 1) отражает ключевые модули Python-сервиса. Модуль-загрузчик выполняет запросы к открытому API маркетплейса и получает данные в формате JSON, после чего модуль-обработчик данных преобразует полученные документы, извлекая необходимые поля (например, сведения о товарах, ценах и остатках) и формируя структурированные объекты. Эти данные передаются в модуль записи в БД, который сохраняет их в локальную базу данных SQLite. Параллельно модуль экспорта метрик агрегирует показатели работы системы, включая количество успешных и ошибочных запросов, время отклика API и объём обновлённых записей, после чего публикует их в формате, совместимом с Prometheus. Для обеспечения устойчивости функционирования предусмотрен модуль регистрации ошибок, фикси-

рующий сетевые сбои и некорректные ответы API, а также модуль-планировщик, управляющий периодичностью опроса API и обеспечивающий регулярное обновление данных. Такая организация компонентов позволяет поддерживать модульность системы и упрощает её сопровождение и расширение.

На уровне классов (рисунок 2) представлены клиент API для взаимодействия с API, парсер продуктов и продукт для обработки данных, обработчик базы данных для операций с БД, сборщик метрик для показателей и планировщик для цикла, связывая бизнес-логику с хранением и мониторингом.

Реализация выполнена на Python с использованием библиотеки requests, многопоточности concurrent.futures и клиента prometheus_client. Данные сохраняются в SQLite, метрики экспортируются в Prometheus и отображаются в Grafana. Такой выбор технологий позволяет локально воспроизвести систему и исследовать её поведение в контролируемых условиях [6; 7].

Использование C4-нотации позволило описать архитектуру на разных уровнях абстракции и связать проектные решения с инженерной реализацией. Практическая часть демонстрирует корректность модели и применимость архитектурных принципов для мониторинга маркетплейсов.

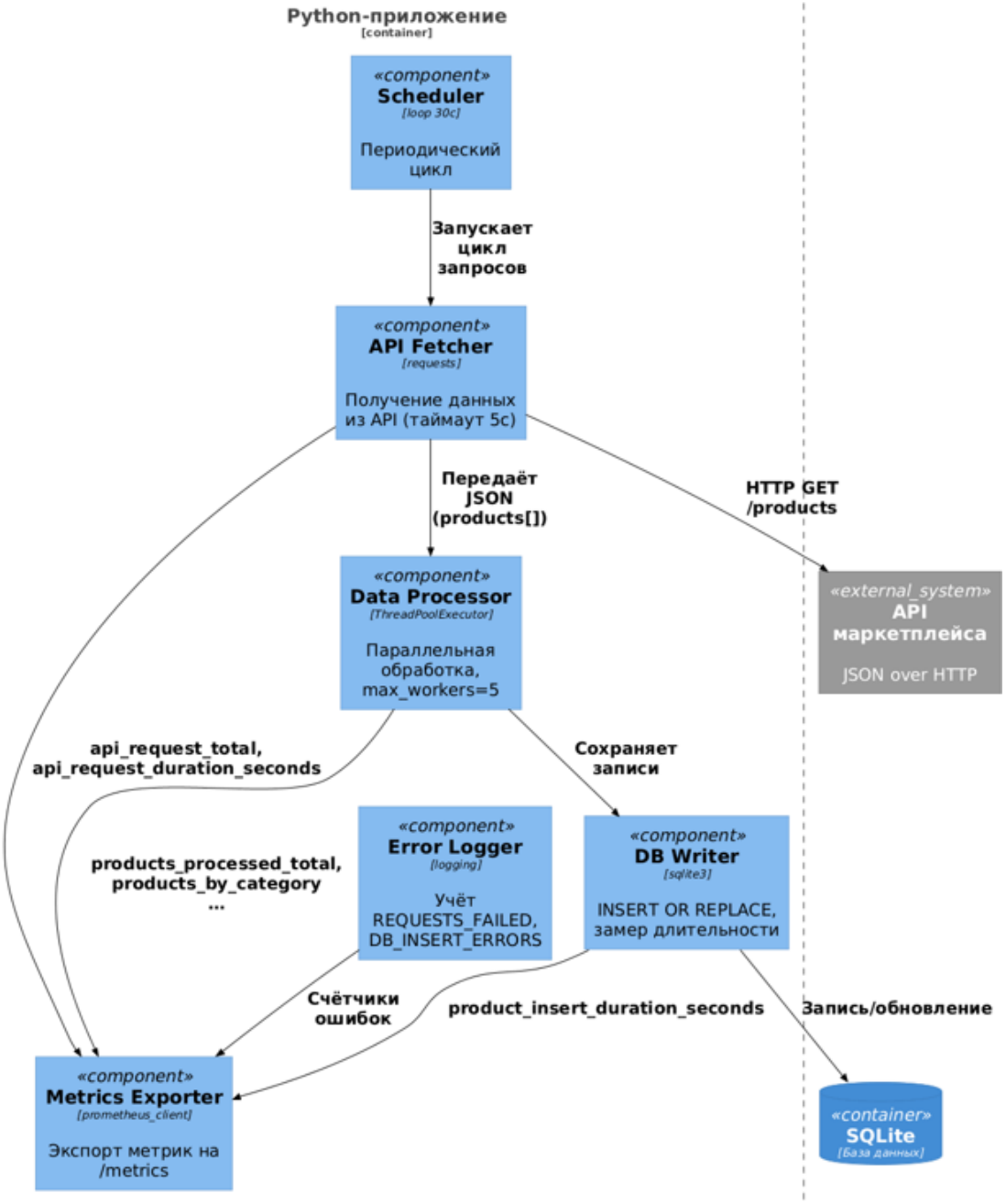

Рисунок 1. Диаграмма компонентов Python-приложения (C3)

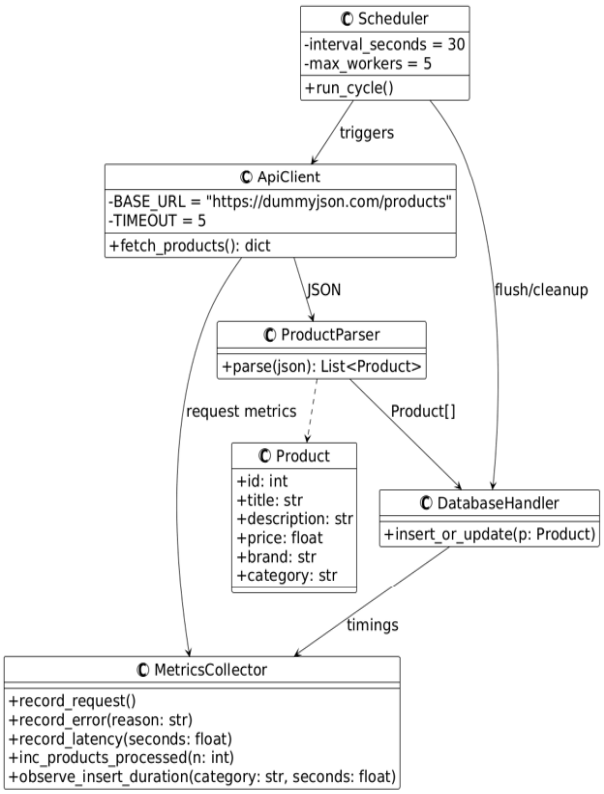

Рисунок 2. Диаграмма классов кода (C4)

Визуализация метрик в Grafana

В рамках системы мониторинга реализован модуль визуализации телеметрических данных с использованием Grafana – инструмента построе-ния интерактивных аналитических панелей на основе временных рядов, собираемых Prometheus. Источником данных служит Prometheus-сервис, опрашивающий Python-сервис с интервалом 15 секунд. Все метрики автоматически экспортируются на HTTP-эндпоинт http://localhost:8000/metrics [1; 7].

На основе собранных метрик создан единый дашборд (рисунок 3), отображающий ключевые показатели работы сервиса в реальном времени: количество API-запросов, ошибок, суммарную и среднюю длительность запросов, обработанные товары, распределение по категориям, ошибки вставки в базу, распределение цен и среднюю длину описаний.

Единый дашборд обеспечивает оперативный мониторинг состояния сервиса и позволяет своевременно выявлять отклонения без анализа необработанных данных. Модуль визуализации является неотъемлемой частью архитектуры мониторинга, поддерживая принятие решений в условиях высоконагруженных e-commerce-систем.

Рисунок 3. Сводный дашборд в Grafana

Анализ полученных метрик

Для оценки работы Python-сервиса использовались модельные данные, имитирующие нагрузку маркетплейса. Период наблюдения составил 30 минут, что позволило накопить репрезентативный объём данных для выявления закономерностей в работе Python-сервиса, ориентированного на обработку информации в маркетплейсе. За этот период зафиксировано 95 API-запросов (область 3) с равномерным ростом от 40 до 100 запросов в минуту (область 10), что свидетельствует о стабильной

нагрузке. Отсутствие ошибок при выполнении запросов (область 4) подтверждает надёжность интеграции компонентов. Средняя длина описания товаров (область 1) составила 118–119 символов, близко к рекомендованным 120, обеспечивая оптимальный баланс информативности и читаемости. Суммарная длительность всех запросов увеличилась с 16 до 40 секунд (область 2), при этом среднее время отклика одного API-запроса оставалось на уровне 0,4 с (область 9), что соответствует допустимым значениям для высоконагруженных систем. Обработано 2850 товаров (область 7), из которых 1440 – категория groceries (область 6), отражая структуру трафика и предпочтения пользователей. Равномерное ценовое распределение (область 5) создаёт предпосылки для динамического ценообразования и персонализации. Отсутствие ошибок при вставке в СУБД (область 8) подтверждает устойчивость базы и корректность валидации данных.

Таким образом, анализ метрик демонстрирует стабильность сервиса, контроль производительности и основу для оптимизации и масштабирования при работе с внешними API.

Моделирование аварийных ситуаций

Для оценки надёжности Python-сервиса было проведено моделирование аварийных ситуаций на ПК с процессором Intel(R) Core(TM) i7-1065G7 CPU @ 1.30–1.50 GHz, 16 ГБ оперативной памяти и 64-разрядной операционной системой Windows 10 Pro версии 22H2. Сценарий включал отказ внешнего API, перегрузку базы данных, временное снижение доступной памяти и процессорных ресурсов, а также увеличение сетевых задержек, что позволило проверить устойчивость сервиса и корректность фиксации сбоев ключевых подсистем: взаимодействия с внешними API, работы базы данных, использования вычислительных ресурсов и сетевых задержек.

Для анализа создан специализированный дашборд в Grafana (рисунок 4). Сценарий сбоя внешнего API (область 1) показал частичную деградацию: успешные запросы – 0,0667 запроса/с, ошибки – 0,0444 запроса/с, подтверждая корректность мониторинга и фиксацию нарушений взаимодействия с внешними сервисами. При росте нагрузки (область 2) среднее время отклика составило 1,31 с, демонстрируя эффективность параллельной обработки данных с помощью потоков (ThreadPoolExecutor).

Мониторинг CPU (область 3) показал низкое использование – 0,000695 секунд на обработку одного запроса, обеспечивая достаточный запас ресурсов. Средняя сетевая задержка (область 4) составила 1,31 с, моделируя перегрузку каналов. Отказы базы данных (область 5) – 3,04 успешных операций/с и 0,289 ошибок/с, подтверждая транзакционную устойчивость. Мониторинг памяти (область 6) показал 62,4 МБ, что отражает постепенное накопление объектов в памяти, имитирующее утечку памяти, как показано на рисунке 4.

Рисунок 4. Дашборд мониторинга аварийных ситуаций

Это позволяет оценить устойчивость сервиса при росте потребления памяти. Дашборд воспроизвёл ключевые сценарии, подтверждая практическую применимость для стресс-тестирования маркетплейса и разработки мер по повышению надёжности.

Заключение

Разработанный Python-сервис обеспечивает контроль ключевых параметров маркетплейса: стабильность API-запросов, производительность вычислений, корректность обработки данных и состояние базы. Автоматизированный сбор и визуализация метрик через Prometheus и Grafana позволяют своевременно выявлять отклонения и поддерживать устойчивость системы без избыточного использования ресурсов. Расширенный набор метрик учитывает характеристики данных и работу критических подсистем, а потоковая обработка с ограничением параллелизма повышает производительность и устойчивость к сбоям. Моделирование аварийных ситуаций подтвердило фиксацию сбоев внешних API, деградации базы, сетевых задержек, роста нагрузки и накопления ресурсов, демонстрируя практическую применимость дашборда для стресс-тестирования и выработки мер повышения надёжности. Подход подчёркивает значимость системного мониторинга в распределённых архитектурах, обеспечивая оперативное реагирование на инциденты и формирование базы для анализа производительности. В дальнейшем сервис может масштабироваться под более крупные маркетплейсы и высоконагруженные системы, сохраняя модульность и адаптивность для задач e-commerce.

REFERENCES

1. Pivotto J, Brazil B. Prometheus: Up & Running : Infrastructure and Application Performance Monitoring. 2nd ed. O'Reilly Media, Incorporated; 2023. 415 p.

2. Salituro E. Learn Grafana 10. x : A Beginner's Guide to Practical Data Analytics, Interactive Dashboards, and Observability. Second edition. Birmingham: Packt Publishing Ltd; 2023. 542 p.

3. Grafana Documentation. Official documentation of Grafana – open platform for analytics and monitoring. Available at: https://grafana.com/docs (accessed: 12.07.2025).

4. Chapman R, Holmes P. Observability with Grafana: Monitor, control, and visualize your Kubernetes and cloud platforms using the LGTM stack. Birmingham: Packt Publishing Ltd; 2023. 356 p.

5. The C4 model for visualising software architecture. Available at: https://c4model.com/ (accessed: 12.07.2025).

6. Turnbull J. Monitoring with Prometheus. Turnbull Press; 2018. 394 p.

7. Python 3.13.7 documentation. Available at: https://docs.python.org/3/ (accessed: 12.07.2025).

PISKUN E., KATSKO E.

PERFORMANCE AND RELIABILITY ANALYSIS OF AN E-COMMERCE PLATFORM MONITORING SYSTEM BASED ON PROMETHEUS AND GRAFANA

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Abstract. *A Python-based monitoring service for user metrics interacting with an open marketplace API has been developed. The architecture is implemented using Prometheus and Grafana and is focused on monitoring the performance of key stages of data processing: the number of requests, errors, response time, database write speed, and product characteristics. To assess system resilience, failure scenarios were simulated, including external API outages, database degradation, network delays, increased load, and memory leaks. The use of stream data processing in combination with SQLite ensures high performance and reliability.*

Keywords: *monitoring, distributed systems, Python service, mid-tier marketplace, Prometheus, Grafana, multithreading, performance, telemetry, stress testing, data processing, API integration*

Пискун Екатерина Сергеевна

Белорусский государственный университет информатики и радиоэлектроники (БГУИР), г. Минск, Республика Беларусь

Кандидат экономических наук. Доцент кафедры проектирования информационно-компьютерных систем БГУИР. Сфера научных интересов: анализ и оптимизация бизнес-процессов, исследование пользовательских и операционных метрик, применение методов системного анализа и визуализации данных для принятия управленческих решений.

Ekaterina Piskun

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

PhD. Associate Professor of the Department of Information and Computer Systems Design at the BSUIR. Research interests: analysis and optimization of business processes, study of user and operational metrics, application of methods of system analysis and data visualization for making management decisions.

E-mail: espiskun@gmail.com

Котько Елизавета Николаевна

Белорусский государственный университет информатики и радиоэлектроники (БГУИР), г. Минск, Республика Беларусь

Ассистент кафедры экономической информатики. Магистрант кафедры проектирования информационно-компьютерных систем БГУИР. Сфера научных интересов: системный анализ информационных систем.

Elizaveta Katsko

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Assistant of the Department of Economic Informatics. Master's student at the Department of Information and Computer Systems Design at the BSUIR. Research interests: systems analysis of information systems.

E-mail: lizakotsko2@gmail.com