Выходные данные полученной модели содержат обработанные изображения, которые являются прогнозами нейронной сети о возможных дефектах, и данная информация передается специалисту для подведения итога о правильности прогноза. В случае достижения неудовлетворительного результата, он будет отправлен в эту же модель на переобучение. Применение нейронных сетей во время мониторинга за объектами ЛЭП позволит увеличить эффективность определения возможных угроз заранее и уменьшит затратность ресурсов на технологическое обслуживание.

СПИСОК ЛИТЕРАТУРЫ

1. Хэ К., Чжан С., Жэнь С., Сун Дж. Глубокое остаточное обучение для распознавания изображений // Труды Конференции IEEE по компьютерному зрению и распознаванию образов (CVPR). – 2016. – С. 770–778. DOI: 10.1109/CVPR.2016.90.
2. Редмон Дж., Диввала С., Гиршик Р., Фархад А. Вы смотрите только один раз: Унифицированное обнаружение объектов в реальном времени // Труды Конференции IEEE по компьютерному зрению и распознаванию образов (CVPR). – 2016. – С. 779–788. DOI: 10.1109/CVPR.2016.91.
3. Джочер Г. и др. YOLOv8: Ultralytics YOLO для обнаружения и сегментации объектов*. Репозиторий GitHub. – URL: https:// github.com/ultralytics/ultralytics. [Дата доступа – 10.11.2024]
4. Тан М., Пан, Р., Ле, К.В. EfficientDet: Масштабируемое и эффективное обнаружение объектов // Труды Конференции IEEE/CVF по компьютерному зрению и распознаванию образов (CVPR). – 2020. – С. 781–790. DOI: 10.1109/CVPR42600.2020.01080.
5. Чжан Ч., Бенджио С., Хардт М., Рехт Б., Виньялс, О. Понимание глубокого обучения (по-прежнему) требует пересмотра обобщения // Коммуникации ACM. – 2021. – Т. 64(3). – С. 107–115. DOI: 10.1145/3446776.
6. Эверингем М., Ван Гоол Л., Уильямс К.К., Уинн,Дж., Зиссман А. Задача визуальных объектов Pascal (VOC) // Международный журнал компьютерного зрения. – 2010. – С. 303–338. DOI: 10.1007/s11263-009-0275-4.
7. Лин Т.-Й., Мэр М., Белонджи С., Хэйс Дж., Перона П., Раманан Д., Доллар П., Цитник К.Л. Microsoft COCO: Общие объекты в контексте // Европейская конференция по компьютерному зрению (ECCV). – 2014. – С. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

РАЗРАБОТКА АСИНХРОННЫХ МИКРОСЕРВИСОВ ДЛЯ ОБРАБОТКИ БЛОКЧЕЙН-ТРАНЗАКЦИЙ В СЕТИ УМНЫХ СЧЁТЧИКОВ

А.Ю. Дёмин[1], Д.И. Дмитрийчук[2], В.А. Зарубин[3]

[1] Томский политехнический университет, ИШИТР, ОИТ
Томский государственный университет, ИПМКН, каф. ТОИ
[2] Томский политехнический университет, ИШИТР, ОИТ, группа 8К11
[3] Томский политехнический университет, ИШИТР, ОИТ, группа 8К13

Научный руководитель: А.Ю. Дёмин, к.т.н., доцент ОИТ ИШИТР ТПУ,
доцент ИПМКН каф. ТОИ ТГУ

Введение

С ростом числа умных счётчиков и их значимости в энергетических сетях вопросы учёта и контроля потребления электроэнергии становятся всё более актуальными. Для эффективного решения этих задач всё чаще используют блокчейн-технологии, которые обеспечивают безопасность, прозрачность и неизменность данных [1]. В рамках таких решений традиционно используется монолитный блокчейн, где все транзакции обрабатываются синхронно и централизованно.

В этом подходе данные от всех счётчиков собираются и обрабатываются последовательно в рамках одной системы, что помогает гарантировать точность и последовательность операций. Однако синхронная обработка и централизованное хранение данных имеют свои ограничения. При увеличении числа устройств возникает проблема масштабируемости, так как нагрузка на систему растёт, что может привести к задержкам в обработке транзакций и снижению производительности.

Современные тенденции в разработке программного обеспечения подталкивают к использованию микросервисной архитектуры, которая позволяет создавать распределённые и масштабируемые системы [2]. Микросервисы становятся всё более популярными благодаря своей гибкости и возможности изолировать функциональность в отдельные компоненты.

В контексте блокчейн-технологий и умных счётчиков применение асинхронных микросервисов открывает новые возможности для повышения эффективности, скорости и масштабируемости обработки данных. Использование этой архитектуры позволяет обрабатывать большое количество транзакций параллельно, что улучшает производительность системы и ускоряет взаимодействие с устройствами в реальном времени.

В данной работе предлагается разработка асинхронной микросервисной системы для обработки блокчейн-транзакций, связанных с данными умных счётчиков, с использованием языка программирования Python и асинхронного веб-фреймворка FastAPI. Такой подход позволит эффективно обрабатывать параллельные запросы и обеспечивать бесперебойную обработку данных в реальном времени.

Цель работы – создать систему, использующую микросервисную архитектуру для обработки транзакций, связанных с учётом потребления электроэнергии, обеспечивая высокую производительность, безопасность и прозрачность с использованием блокчейн-технологий.

Микросервисная архитектура и её применение

Микросервисная архитектура представляет собой подход к разработке программного обеспечения, при котором система разбивается на независимые, самодостаточные компоненты – микросервисы. Каждый микросервис выполняет одну конкретную задачу и взаимодействует с другими через чётко определённые интерфейсы, такие как API или сообщения. Это позволяет разрабатывать, тестировать и масштабировать каждый компонент системы отдельно, что значительно повышает гибкость и ускоряет процесс разработки [3].

Микросервисы могут работать автономно, но для полноценного функционирования системы важно, чтобы они могли эффективно взаимодействовать между собой. Для этого часто применяются асинхронные коммуникации, где данные передаются с использованием очередей сообщений или событий, что позволяет избежать блокировок и улучшить производительность системы. Такой подход идеально подходит для систем с высокой нагрузкой, таких как обработка блокчейн-транзакций, где требуется быстрое и параллельное выполнение множества операций.

Используя микросервисы в контексте записи транзакций с умных счётчиков, можно эффективно распределить задачи блокчейна по независимым узлам. Разделение позволит различным элементам системы работать параллельно, без каких-либо задержек. Данная архитектура позволит существенно повысить масштабируемость и отказоустойчивость системы.

Разработка микросервисов

Проект состоит из трёх основных микросервисов, каждый из которых выполняет свои задачи и взаимодействует с другими сервисами через REST API, а также общую базу данных PostgreSQL. Использование Docker-контейнеров и Docker-compose помогает легко разворачивать проект, управлять микросервисами, обернутыми в контейнеры, что позволяет облегчить масштабирование системы, ее поддержку и, в перспективе – обновление.

Сервис сбора данных (Data Collector Service) получает данные с внешних устройств – счетчиков или API, затем записывает их в базу данных. Сервис для записи данных в цепочку блокчейна (Blockchain Service) необходим для хранения данных в неизменяемой структуре блоков блокчейна, сводя к минимуму возможность подделки данных и позволяя контролировать изменения в них. Сервис управления базой данных и миграциями взаимодействует с базой данных и управляет миграциями базы данных с использованием Alembic, необходим для автоматизации процесса обновления структуры БД, сборки и просмотра цепочки блойкчейна.

Все микросервисы используют базу данных PostgreSQL для обмена данными. В базе данных хранятся промежуточные данные, полученные с внешних устройств. Затем блокчейн-сервис выгружает данные из БД, проверяя их целостность перед инициализацией нового блока, создает новый блок и фиксирует его в таблице blockchain базы данных.

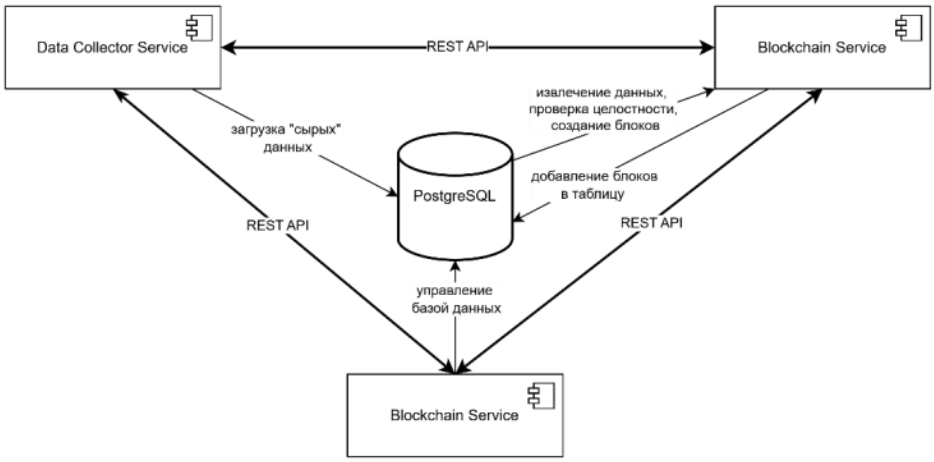

Рис. 1. Схема взаимодействия микросервисов на узле-счетчике

Для обеспечения целостности структуры базы данных используется Database Service, который применяет миграции при запуске. Также данный сервис необходим для обеспечения доступа к базе с целью получения истории потребления электроэнергии конкретным счетчиком, а также для просмотра всей цепочки блоков, записанных в таблицу blockchain.

Для организации единой точки доступа к микросервисам, облегчения маршрутизации запросов, распределения нагрузки между несколькими экземплярами микросервисов, а также централизованного управления системой используется API Gateway. Это решение помогает снизить нагрузку на микросервисы, повысить производительность и упростить мониторинг запросов.

Для развёртывания системы и оборачивания микросервисов с контейнеры используется Docker и Docker Compose. Конфигурация каждого сервиса описана в Dockerfile, а настройки запуска кластера подготовленных контейнеров – в docker-compose.yml

Тестирование системы

Для проверки корректности работы микросервисов и устойчивости к сбоям было проведено тестирование системы с помощью фреймворка pytest. Тесты покрывают основные функции микросервисов, такие как: добавление и валидацию блоков в блокчейне, сбор данных с устройств, управление миграциями и проверку состояния базы данных, а также взаимодействие через REST API.

Тесты блокчейн-сервиса проверяют создание блоков и успешное получение цепочки блокчейна, задавая в качестве входных данных подменённые хэш-суммы предыдущих блоков. После чего система правильно помечала цепочку невалидной. Эти тесты позволяют удостовериться в неизменности данных и корректности работы алгоритма. В тестах сервиса-сборщика данных проверялся процесс сбора данных с устройств и запись информации в базу данных, пройденные тесты доказывают актуальность и точность получаемой информации. Микросервис базы данных проверял работу миграций и обновления статуса базы данных, подтвердив стабильность работы приложения.

Асинхронность тестов, написанных с помощью модулей AsyncClient и pytest.mark.asyncio, помогает моделировать приближенную к реальной нагрузку и выполнять параллельные опе-

рации, чтобы выявлять потенциальные ошибки и сбои. Таким образом были проверены компоненты микросерсивов, а также взаимодействие между ними, что показало устойчивость работы системы в условиях, близких к реальным.

Результаты тестов показали высокое покрытие кода – более 90 %, что свидетельствует о комплексной проверке системы и минимизации вероятности сбоев при работе в условиях реальных данных.

Заключение

В результате работы была разработана система, состоящая из трех приложений, организованных в рамках микросервисной архитектуры. Каждый из микросервисов выполняет свою задачу: сервис сбора данных принимает и записывает информацию с различных устройств в базу данных, сервис блокчейна обеспечивает сохранность и неизменность данных, записывая их в цепочку блоков, а сервис управления базой данных отвечает за миграции и стабильную работу с данными.

Все микросервисы взаимодействуют через общую базу данных PostgreSQL, это позволяет каждому сервису эффективно обмениваться информацией, записывая и извлекая данные из единого хранилища. Сервис сбора данных сохраняет данные в базе, к которым впоследствии обращается сервис блокчейна для обработки данных и добавления их в цепочку блоков.

Для упрощения взаимодействия между этими микросервисами и централизованного управления запросами был внедрён API Gateway. Это решение не только упростило маршрутизацию, но и улучшило мониторинг и управление запросами.

Запуская данную программу на каждом счетчике, достигается децентрализованность системы, где каждый счетчик выполняет свою роль как независимый узел, что позволяет системе продолжать функционировать даже в случае выхода из строя отдельных компонентов.

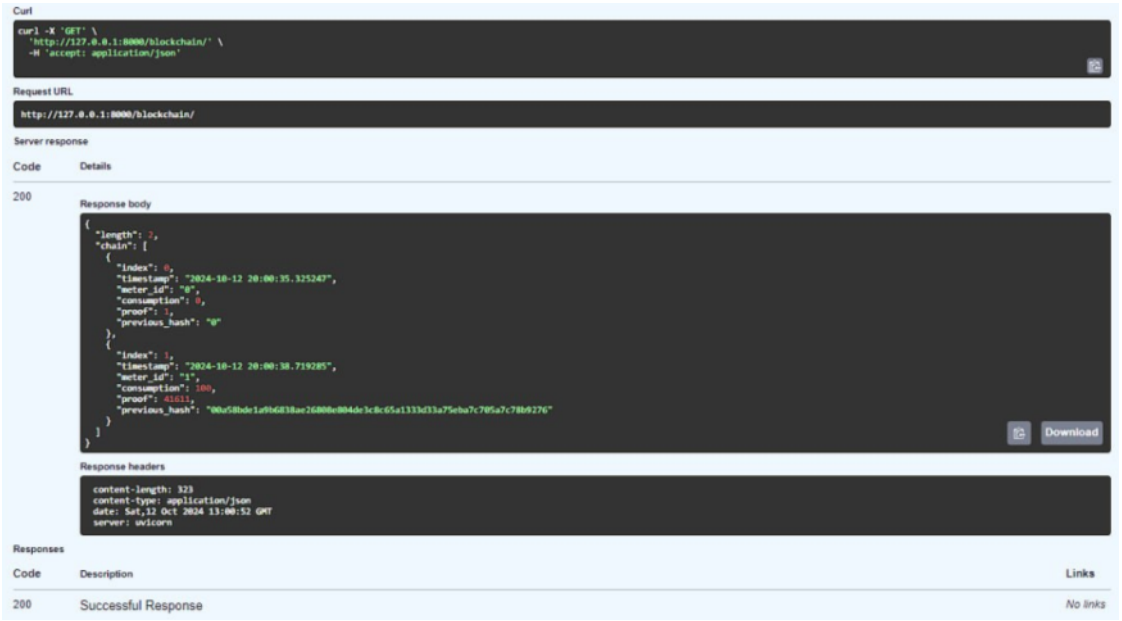

Рис. 2. Результат работы database_service при получение общей цепи блокчейна

СПИСОК ЛИТЕРАТУРЫ

1. BSEMS–A Blockchain-Based Smart Energy Measurement System / M. Sing, S. Ahmed, S. Sharma, S. Singh, B. Yoon // Sensors. – 2023. – V. 23.
2. Микросервисы (Microservices) // habr.com 2015. – URL: https:// habr.com/ru/articles/249183/ (дата обращения 02.11.2024).
3. Взаимодействие в архитектуре микросервисов // habr.com 2022. – URL: https:// habr.com/ru/companies/slurm/articles/675682/ (дата обращения 02.11.2024).