4. Козлов, О.Л. Применение машинного обучения для оптимизации ИПС // Цифровизация производства: вызовы и решения: материалы II Всероссийской научно-практической конференции. -Москва, -2023. -С. 112-119.

5. Смирнов, С.А. Цифровые двойники в машиностроении. -Москва : Инфра-М, -2024. -415 с.

6. Фёдоров, А. И. Виртуальное моделирование как основа киберфизических систем // Промышленная инженерия. -2023. -Т. 12, -№ 4. -С. 60-72.

7. Ковалев, Д.С. Разработка математических моделей для виртуального двойника // Актуальные вопросы цифровизации производства: материалы V Всероссийской научно-практической конференции. -Санкт-Петербург, -2022. -С. 88-95.

МОНИТОРИНГ ПРОИЗВОДИТЕЛЬНОСТИ ВЫСОКОНАГРУЖЕННЫХ СИСТЕМ НА ОСНОВЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

[1]Халиляев Л.С., [1]Нодиржонов Р.Н., [2]Медведева О.А.
[1]Студент кафедры точных наук и информационных технологий
Филиал КФУ в г. Джизак, Республика Узбекистан
[2]К.ф.-м.н., доцент кафедры информационных систем
Казанский федеральный университет, г. Казань, Россия
OAMedvedeva@kpfu.ru

Аннотация: В рамках данного исследования осуществлена разработка микросервисного приложения с использованием двух различных подходов: традиционного и реактивного программирования с применением фреймворка Spring. Проведены нагрузочные тесты, которые позволяют оценить, какой из подходов лучше соответствует требованиям разработки высоконагруженных систем.

Ключевые слова: микросервисы, нагрузка, Spring, Java, мониторинг высоконагруженных систем.

PERFORMANCE MONITORING OF MICROSERVICE-BASED SYSTEMS

[1]Khalilyaev L.S., [1]Nodirjonov R.N., [2]Medvedeva O.A.
[1]Student, Department of Exact Sciences and Information Technology
Branch of KFU, Jizzakh, Uzbekistan
[2]PhD Associate Professor Department of Information Systems KFU, Kazan, Russia
OAMedvedeva@kpfu.ru

Abstract: Within the framework of this study, a microservice application was developed using two different approaches: traditional and reactive programming using the Spring framework. Load tests have been conducted to assess which approach best meets the requirements of high-load systems.

Keywords: microservices, performance, Spring, Java, high-load systems monitoring.

Проблема обеспечения высокой производительности веб-приложений остается одной из наиболее важных в современной ИТ-индустрии. Особое внимание этот вопрос приобретает в контексте высоконагруженных систем, где одновременно обрабатываются запросы тысяч или даже миллионов пользователей. Постоянный рост интернет-аудитории и переход ключевых сервисов – от банкинга и электронной коммерции до дистанционного образования – в цифровую среду предъявляют исключительные требования к отказоустойчивости, скорости отклика и стабильности работы программного обеспечения.

В данной статье рассмотрим комплексное исследование производительности веб-приложения, разработанного на языке Java с использованием фреймворка Spring. Выбор этого технологического стека не случаен: Java, благодаря своей кроссплатформенности и мощной экосистеме, является одним из лидеров в создании корпоративных и высоконагруженных приложений. Фреймворк Spring, в свою очередь, стал де-факто отраслевым стандартом для построения сложных, масштабируемых и легко тестируемых систем на Java, предлагая богатейший набор инструментов для всех уровней приложения – от управления зависимостями и транзакциями до безопасности и взаимодействия с базами данных.

Актуальность данного исследования многогранна. Во-первых, она обусловлена всеобщей цифровизацией и тем, что информационные технологии проникли во все сферы жизни. Во-вторых, и это является ключевым фокусом, – взрывным ростом значимости образовательных платформ в рамках современной парадигмы непрерывного обучения. Для таких систем производительность напрямую конвертируется в качество образовательного процесса. Задержка в загрузке лекционных материалов, медленная работа интерактивных заданий или недоступность платформы в час пик создают барьеры для обучения, снижая мотивацию студентов и эффективность всего учебного процесса. Таким образом, цель нашего исследования заключается в том, чтобы не просто измерить абстрактные метрики производительности, а выявить узкие места и предложить конкретные методы оптимизации, направленные на максимальное ускорение доступа пользователей к образовательному контенту и их персональным данным. Успешное решение этой задачи позволит повысить стабильность и отзывчивость образовательных платформ, что в конечном итоге внесет вклад в улучшение и демократизацию современного образования.

Высоконагруженные приложения представляют собой сложные распределенные системы, спроектированные для обработки огромного количества одновременных запросов при сохранении строгих требований к задержке, доступности и отказоустойчивости. Такие системы, например, социальные сети, торговые площадки или образовательные платформы, требуют применения специализированных архитектурных паттернов, поскольку традиционные решения не справляются. В качестве объекта исследования выступает приложение, разработанное на основе микросервисной архитектуры. Ключевое отличие этого подхода заключается в принципе декомпозиции: вместо единой, тесно связанной кодовой базы приложение разделяется на набор небольших, слабосвязанных и независимо развертываемых сервисов. Каждый такой сервис инкапсулирует определенную бизнес-функцию (например, «аутентификация пользователей», «управление курсами», «обработка данных») и взаимодействует с другими через четко определенные API, чаще всего по протоколу HTTP или с помощью асинхронных сообщений. Эта модульность обеспечивает принципиально иной уровень устойчивости: отказ одного сервиса не приводит к полной потери работоспособности всей системы, а лишь к временной недоступности конкретной функции.

Выделим преимущества микросервисной архитектуры для высоконагруженных систем:

- команды разработчиков могут независимо разрабатывать, тестировать и развертывать свои сервисы, что значительно ускоряет вывод нового функционала;

- вместо масштабирования всего приложения целиком можно увеличить вычислительные ресурсы только для тех сервисов, которые испытывают пиковую нагрузку (например, сервис аутентификации во время часа-пик), что ведет к значительной оптимизации затрат;

- различные сервисы могут быть реализованы на разных языках программирования и с использованием разных технологий хранения данных, что позволяет выбрать наиболее подходящий инструмент для конкретной задачи.

Для объективной оценки эффективности архитектуры проведены комплексные нагрузочные тесты. Их задача выходит далеко за рамки простого определения максимальной пропускной способности. Современное нагрузочное тестирование решает следующие критически важные задачи:

- выявление компонентов, которые первыми выходят из строя под нагрузкой (база данных, отдельный микросервис, брокер сообщений);

- анализ поведения системы при сбоях, т.е. как система восстанавливается после отказа одного из сервисов или базы данных;

- проверка механизмов устойчивости: эффективность таких паттернов, как Circuit Breaker, Retry и Fallback.

- прогнозирование производительности: построение моделей, позволяющих предсказать, как система будет вести себя при дальнейшем росте нагрузки.

Архитектура анализируемого приложения, представлена на рисунке 1 и включает следующие ключевые компоненты:

- набор микросервисов: каждый сервис реализован на Java с использованием фреймворка Spring Boot, который предоставляет готовые решения для инъекции зависимостей, создания REST API и работы с данными.

- база данных: в качестве основного хранилища используется PostgreSQL – надежная реляционная СУБД с открытым исходным кодом, обеспечивающая строгую целостность данных и

поддержку транзакций. Альтернативами с аналогичными гарантиями являются MS SQL Server, MySQL или Oracle.

- синхронное взаимодействие: для прямых запросов-ответов между сервисами применяются синхронные HTTP-вызовы (часто через клиент с балансировкой нагрузки, такой как Spring Cloud LoadBalancer).

- асинхронное взаимодействие: для обработки фоновых задач и событийной коммуникации задействован брокер сообщений Apache Kafka. Это позволяет развязать сервисы во времени: производитель сообщения не должен ждать, пока потребитель его обработает, что повышает общую отзывчивость системы.

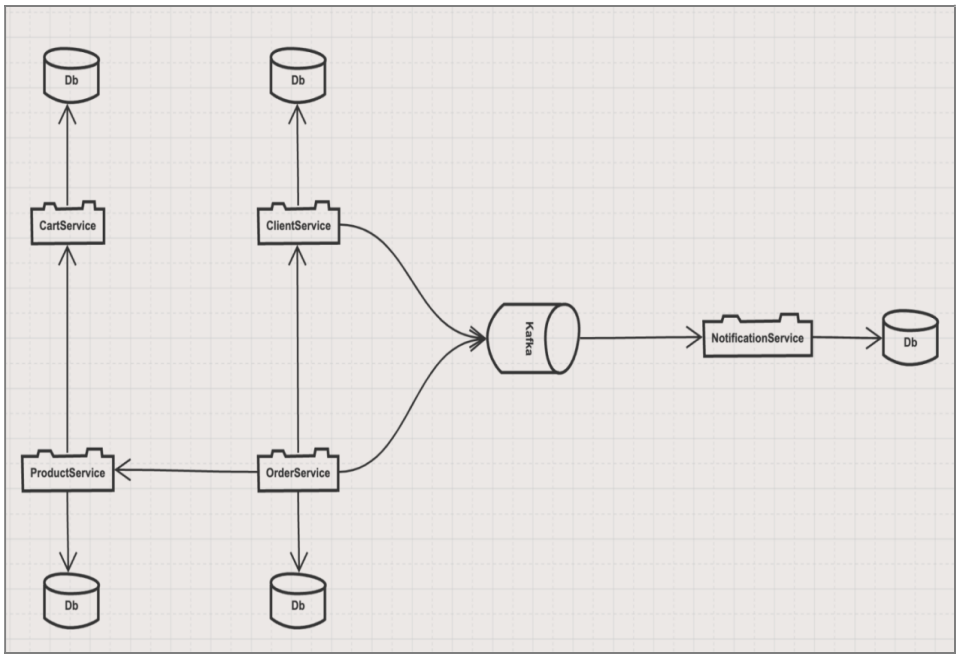

Рис. 1. Архитектура приложения

Проведем сравнительный анализ подходов к разработке высоконагруженных приложений. Каждый из рассмотренных подходов к разработке обладает уникальными преимуществами и недостатками, определяющими область его применения.

1. Традиционное (императивное) программирование. Суть подхода заключается в том, что обработка запросов по модели «один поток на запрос». Каждый клиентский запрос выполняется в отдельном системном потоке, что при высокой нагрузке быстро приводит к исчерпанию ресурсов и блокировкам. Преимуществом подхода является простота разработки, понимания и отладки, широкая распространенность технологий и кадров, что снижает стоимость и время реализации небольших проектов. К недостаткам относится низкая производительность и отказоустойчивость под высокой нагрузкой из-за высоких накладных расходов на системные потоки и синхронных блокирующих операций.

2. Реактивное программирование. Суть подхода состоит в использовании неблокирующих операций и фиксированного числа потоков для асинхронной обработки множества запросов, позволяет эффективно использовать ресурсы центрального процессора и памяти. Преимущества: высокая пропускная способность и эффективность утилизации ресурсов, что идеально подходит для систем с большим количеством одновременных подключений. Недостатки: высокий порог входа для разработчиков, требующий знаний в асинхронном программировании, усложненная отладка и поддержка кода из-за его событийно-ориентированной природы.

3. Виртуальные потоки (Java 21+). Суть подхода состоит в том, что виртуальные потоки, которые работают поверх обычных системных потоков, позволяют сохранить простую модель «поток на запрос», но без высоких накладных расходов. Преимущества подхода: значительный прирост производительности для блокирующих операций (ввод-вывод), позволяет сочетать производительность, близкую к реактивному подходу, с простотой разработки, характерной для традиционного стиля. Недостатки: наибольшая эффективность достигается при модернизации традиционных приложений; в чисто реактивных стеках преимущества могут быть менее выраженными, так как они изначально минимизируют блокировки.

Выбор архитектуры зависит от конкретных задач. Традиционный подход экономичен для простых систем, реактивный – для максимальной производительности в условиях асинхронного ввода-вывода, а виртуальные потоки предлагают компромиссный вариант, значительно повышая производительность традиционных приложений с минимальными изменениями в коде.

Проведенный анализ нагрузочного тестирования приложений, разработанных с помощью вышеописанных подходов, позволяет сделать следующие выводы. Приложение, разработанное традиционным способом, имеет низкую производительность и высокую утилизацию ресурсов процессора, но легкая и недорогая стоимость разработки делает этот подход востребованным для небольших и средних проектов, в которых нет необходимости в обработке большого количества одновременных запросов. С помощью переключения приложения, разработанного традиционным способом, на виртуальные потоки, можно достичь хорошего прироста в производительности без удорожания разработки и серверов. Реактивное программирование имеет лучший результат, по сравнению с традиционным подходом, благодаря неблокирующим операциям и отсутствии необходимости создания дополнительного потока на каждый запрос пользователя. Это делает его хорошим вариантом для разработки приложений, в которых есть необходимость в обработке большого количества одновременных запросов к приложению и в команде присутствуют разработчики высокого уровня. Виртуальные потоки в реактивном приложении незначительно улучшают его производительность, особенно если есть большое количество блокирующих операций, которые необходимо выполнять в отдельном потоке.

Высоконагруженные приложения широко применяются в образовании при проектировании и разработке цифровых образовательных ресурсов и онлайн-курсов для студентов и обучающихся, а также, корпоративных платформ для проведения курсов повышения квалификации преподавателей и сотрудников образовательной организации. Такие приложения предоставляют пользователям быстрый доступ к информации, возможность обрабатывать транзакции в режиме реального времени, и обеспечивать надежную и безопасную работу системы.

Список литературы

1. Бевзенко С.А. Процесс автоматизации нагрузочного тестирования веб-систем: основные аспекты // Universum: технические науки: электрон. научн. журн. 2023. 8(113). URL: https://7universum.com/ru/tech/archive/item/15829 (дата обращения: 03.09.2025).

2. Бевзенко С.А. Исследование эффектов нагрузочного тестирования на производительность и надежность системы // Universum: технические науки : электрон. научн. журн. 2023. 9(114). URL: https://7universum.com/ru/tech/archive/item/16001 (дата обращения: 05.09.2025).

3. Документация по виртуальным потокам Java // docs.oracle.com. 2024. URL: https://docs.oracle.com/en/java/javase/21/core/virtual-threads.html (дата обращения: 13.10.25).

4. Реализация распределенной системы, управляемой событиями // Желтовский К. Д., 2021 г., 82 с.

5. Ричардсон Крис. Микросервисы. Паттерны разработки и рефакторинга. Издательство: Питер, 2024. 544 с.

6. Рудометкин В.А. Проектирование высоконагруженных систем. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 79-86. DOI: 10.15514/ISPRAS–2020–32(6)–6

7. Филисов Д.А. Архитектура высоконагруженных приложений // Universum: технические науки: электрон. научн. журн. 2023. 10(115). URL: https://7universum.com/ru/tech/archive/item/16138 (дата обращения: 23.10.2025).

8. Хаматянов М.И. Исследование производительности высоконагруженных систем на основе фреймворка Spring / М.И. Хаматянов, О.А. Медведева // V Международный форум по математическому образованию, посвященный 220-летию Казанского университета (IFME' 2024) [Электронный ресурс]: материалы IV Международного научного семинара “Digital Technologies for Teaching and Learning” (Казань, 25-30 марта 2024 г.) – Казань: Издательство Казанского университета, 2024. – С. 398-403.

9. Хаматянов М.И. Проектирование микросервисов для анализа подходов к разработке высоконагруженных систем на основе фреймворка Spring / Хаматянов М.И., Медведева О.А. // Математическое моделирование процессов и систем: материалы XIII Международной молодежной научно-практической конференции. - Стерлитамакский филиал УУНиТ, 2023. - С. 678-682.

ARTIFICIAL INTELLIGENCE FOR SUSTAINABLE ARCHITECTURE: SMART GREEN BUILDINGS IN UZBEKISTAN

Sorokin S.V.
Senior Lecturer, Westminster International University in Tashkent (WIUT),
Tashkent, Republic of Uzbekistan
sorokinsergei7789@gmail.com
Nailev T.R.
Student, Westminster International University in Tashkent (WIUT), Tashkent,
Republic of Uzbekistan
timur.nailyevv@gmail.com

Annotation: Uzbekistan, a leading Central Asian country in terms of digital transformation and AI adoption, is embracing intelligent technologies to enhance the sustainability of its built environment. This study explores the application of artificial intelligence in the design and energy optimization of green buildings across urban centers such as Tashkent, Samarkand, and Bukhara. By leveraging machine learning, computer vision, and IoT-integrated data streams, this research demonstrates how AI contributes to reducing energy consumption, optimizing HVAC performance, and supporting real-time energy monitoring. Through a case study methodology combined with predictive modeling and empirical evaluation, the study illustrates a comprehensive AI-driven framework for smart sustainable buildings tailored to Uzbekistan's climate, urban density, and energy infrastructure.

Keywords: Smart buildings, artificial intelligence, energy efficiency, green design, Uzbekistan, sustainability, IoT, climate-adaptive architecture

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ ДЛЯ УСТОЙЧИВОЙ АРХИТЕКТУРЫ: УМНЫЕ ЗЕЛЕНЫЕ ЗДАНИЯ В УЗБЕКИСТАНЕ

Сорокин С.В.
Старший преподаватель, Международный Вестминстерский Университет в Ташкенте,
г. Ташкент, Республики Узбекистан
sorokinsergei7789@gmail.com
Наильев Т.Р.
Студент, Международный Вестминстерский Университет в Ташкенте,
г. Ташкент, Республики Узбекистан
timur.nailyevv@gmail.com

Аннотация: Узбекистан, ведущая страна Центральной Азии по уровню цифровой трансформации и внедрения искусственного интеллекта (ИИ), внедряет интеллектуальные технологии для повышения устойчивости своей застройки. В данном исследовании рассматривается применение искусственного интеллекта (ИИ) в проектировании и оптимизации энергопотребления экологичных зданий в таких городских центрах, как Ташкент, Самарканд и Бухара. Используя машинное обучение, компьютерное зрение и интегрированные с Интернетом вещей потоки данных, данное исследование демонстрирует, как ИИ способствует снижению энергопотребления, оптимизации производительности систем отопления, вентиляции и кондиционирования воздуха (ОВК) и поддержке мониторинга энергопотребления в режиме реального времени. Используя методологию тематического исследования в сочетании с предиктивным моделированием и эмпирической оценкой, исследование иллюстрирует комплексную структуру на основе ИИ для создания «умных» устойчивых зданий, адаптированную к климатическим условиям, плотности городской застройки и энергетической инфраструктуре Узбекистана.

Ключевые слова: Умные здания, искусственный интеллект, энергоэффективность, экологичный дизайн, Узбекистан, устойчивое развитие, Интернет вещей, адаптируемая архитектура к изменению климата.

BARQAROR ARXITEKTURA UCHUN SUN’IY INTELTEKT: O‘ZBEKISTONDAGI AQLLI YASHIL IMORATLAR