МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ KVAZAR ONLINE

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Вавилова Андрея Сергеевича

Научный руководитель
доцент, к. ф.-м. н _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

Молекулярное моделирование является одним из наиболее востребованных инструментов современной науки, находящим применение в широком спектре областей — от материаловедения и разработки новых лекарственных препаратов до нанотехнологий и биоинженерии. Возможность предсказывать свойства веществ и поведение молекулярных систем на уровне атомов без проведения дорогостоящих и трудоемких физических экспериментов открывает колоссальные перспективы для научных исследований и промышленных разработок. С ростом вычислительных мощностей и развитием алгоритмов методы молекулярного моделирования становятся все более точными и доступными.

В настоящее время существует множество программных пакетов для молекулярного моделирования, однако они зачастую требуют от пользователя глубоких знаний в области вычислительной химии и физики, а также установки специализированного программного обеспечения на локальные машины. Разработка веб-приложений для научных вычислений позволяет сделать эти инструменты более доступными. Данная дипломная работа посвящена разработке ключевых компонентов такой системы для открытого многопроцессорного программного комплекса молекулярного моделирования Kvazar. Kvazar — это отечественное десктоп-приложение, разработанное сотрудниками Саратовского государственного университета им. Н. Г. Чернышевского.

Актуальность данной дипломной работы обусловлена необходимостью создания эффективной и надежной инфраструктуры для веб-ориентированных научных вычислений. В частности, актуальной задачей является проектирование и реализация серверной части, способной обрабатывать запросы пользователей, управлять вычислительными задачами и обеспечивать взаимодействие между различными компонентами распределенной системы. Использование микросервисной архитектуры и асинхронного обмена сообщениями через брокеры, такие как Apache Kafka, является современным подходом к построению масштабируемых и отказоустойчивых приложений, особенно в контексте интеграции высокопроизводительных вычислительных модулей, написанных на C++, с веб-сервисами.

Целью настоящей дипломной работы является разработка и интеграция API-микросервиса на Spring Boot и вычислительного микросервиса на C++ с использованием Apache Kafka для обеспечения функционирования веб-прило-

жения молекулярного моделирования Kvazar Online.

Для достижения поставленной цели в рамках данной работы были решены следующие **задачи**:

1. Провести анализ существующих подходов к построению распределенных вычислительных систем и обмена сообщениями в них.
2. Спроектировать архитектуру взаимодействия между API-микросервисом и вычислительным микросервисом с использованием Apache Kafka.
3. Разработать API-микросервис на платформе Spring Boot, отвечающий за прием и валидацию данных от фронтенд-части, а также за отправку и получение задач через Apache Kafka.
4. Разработать клиентскую часть вычислительного микросервиса на языке C++, включающую реализацию Kafka-клиента для получения вычислительных задач из брокера и отправки результатов обратно.
5. Реализовать механизмы сериализации и десериализации данных для обмена сообщениями между сервисами с использованием Apache Avro.
6. Обеспечить интеграцию и корректное взаимодействие разработанных компонентов (API-сервиса и C++ Kafka-клиента) через брокер сообщений Apache Kafka.

Теоретическая значимость заключается в освоении современных подходов к разработке распределенных систем и микросервисов. Реализация проекта способствовала углублению понимания архитектуры клиент-серверных приложений.

Практическая значимость состоит в реализации серверной части веб-приложения «Kvazar Online» для молекулярного моделирования. Разработанное решение способствует повышению удобства взаимодействия с системой.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и четырех приложений. Общий объем работы — 55 страниц, из них 42 страницы — основное содержание, включая 4 рисунка, цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Описание предметной области и используемых технологий» представляет обзор теоретических основ и технологического стека, использованного при разработке.

В первом подразделе дается определение молекулярного моделирования как области вычислительной науки, описываются его ключевые идеи, цели и возможности. Рассматриваются различные уровни детализации моделей, от квантово-механических до крупнозернистых, и поясняется фокус данной работы на классических и крупнозернистых методах.

Далее рассматриваются конкретные методы, поддержка которых реализована в приложении: REBO (Reactive Empirical Bond-Order), AIREBO, Tight-Binding и Martini. Для каждого метода кратко описана его суть, область применения и ключевые особенности, что обосновывает необходимость предоставления пользователю выбора инструмента для своих исследований.

Описан фреймворк Spring Boot как расширение Spring, упрощающее создание производительных Java-приложений. Основной концепцией Spring является Inversion of Control (IoC). Она заключается в том, что объектами приложения управляет не разработчик, а Spring IoC Container. Автоматическое управление объектами достигается путем использования Dependency Injection (DI) — особого подхода предоставления зависимостей. Во время поиска и инициализации пользовательских объектов фреймворк внедряет в них ссылки на требуемые сущности.

Преимуществами использования Spring Boot являются: автоконфигурация, встроенные стартеры и серверы.

В качестве брокера сообщений выбран и рассмотрен брокер Apache Kafka. Описаны принципы его работы как распределенной publish-subscribe системы, а также основные понятия: топик как логический канал сообщений, партиции и группы как инструменты масштабирования. Kafka предоставляет три семантики доставки сообщений и идеально подходит для создания событийно-ориентированных, отказоустойчивых и масштабируемых систем.

Для взаимодействия с Kafka из разработанных сервисов были выбраны соответствующие библиотеки. Для Java-сервиса рассмотрены официальная библиотека kafka-clients и высокоуровневая абстракция Spring for Kafka, которая значительно упрощает интеграцию благодаря шаблонам и аннотациям. Она ос-

нована на официальной библиотеке, но «из коробки» реализует многие шаблонные подходы к работе к брокером: есть готовые классы с циклами опроса топиков, особыми видами обмена сообщениями. Это уменьшает количество шаблонного кода.

Для сервиса на C++ рассмотрены библиотеки librdkafka и modern-cpp-kafka. Первая является стандартной, но устаревшей официальной реализацией в функциональном стиле. Вторая — более современная обертка в объектно-ориентированном стиле, который проще для понимания современным разработчиком.

Важнейшим аспектом межсервисной коммуникации является формат данных. В работе был выбран Apache Avro — система бинарной сериализации данных. Описаны его преимущества: компактность, высокая скорость и строгая типизация на основе схем в формате JSON.

Для интеграции Avro с C++ приложениями Apache предоставляет официальную библиотеку avro-cpp. Эта библиотека позволяет C++ программам парсить Avro-схемы, генерировать C++ классы, соответствующие этим схемам, а также выполнять сериализацию C++ объектов в бинарный формат Avro и десериализацию бинарных данных обратно в C++ объекты.

Spring Kafka легко интегрируется с Avro-сериализаторами и десериализаторами, предоставляемыми Confluent, которые не только обрабатывают формат Avro, но и взаимодействуют с Confluent Schema Registry для управления схемами. Аналогично C++, для Java также генерируются классы из Avro-схем с помощью плагинов для системы сборки Maven.

Для централизованного управления схемами Avro использовался Confluent Schema Registry. Он обеспечивает версионирование схем, проверку их совместимости при регистрации новых версий и позволяет значительно уменьшить размер Kafka-сообщений, так как сама схема не передается в каждом сообщении, а передается лишь ее идентификатор.

Описаны основные концепции Docker и Docker Compose: образы и контейнеры, а также Dockerfile для декларативного описания сборки образов. Показано, как Docker Compose позволяет с помощью одного YAML-файла определить и запустить все многокомпонентное приложение, включая сервисы, брокер Kafka и другие зависимости, что критически важно для упрощения разработки, тестирования и развертывания.

Таким образом, в первом разделе обоснован выбор технологического стека для разработки приложения на основе микросервисной архитектуры, включающего интеграцию сервисов, написанных на разных языках программирования.

Второй раздел «Разработка серверной части системы» описывается процесс проектирования и реализации ключевых компонентов приложения.

Раздел начинается с описания общей архитектуры приложения Kvazar Online, состоящей из клиентской части, API-сервиса на Spring Boot, брокера сообщений Apache Kafka, вычислительного сервиса на C++ и реестра схем Schema Registry. Описана общая логика взаимодействия: пользователь отправляет запрос на API-сервис, который преобразует его в задачу и отправляет в Kafka. Вычислительный сервис получает задачу, выполняет расчет и отправляет результат обратно через Kafka. API-сервис принимает результат и возвращает его пользователю. Схема архитектуры системы приведены на рис. 1.

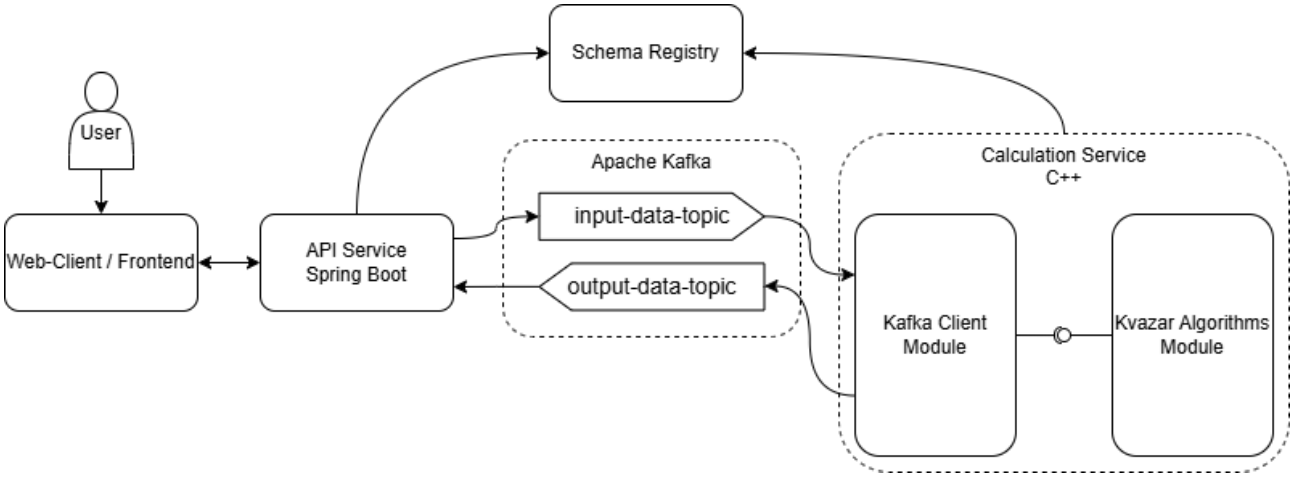

Рисунок 1 – Архитектура приложения

Далее рассматривается реализация API-сервиса на Spring Boot. Его архитектура построена по классическому многослойному принципу. Контроллерный слой, реализованный с помощью Spring MVC, содержит один эндпоинт `/calculate`, принимающий HTTP POST-запросы с JSON-объектом, описывающим параметры расчета. Контроллер работает в асинхронном режиме, что позволяет не блокировать потоки веб-сервера на время ожидания ответа от вычислительной системы.

Сервисный слой инкапсулирует основную бизнес-логику: валидацию входящих данных, преобразование DTO-объекта запроса во внутренний Avro-объект и взаимодействие с Kafka. Для отправки запроса и получения ответа используется `ReplyingKafkaTemplate` из библиотеки Spring for Kafka. Этот компонент автоматически управляет корреляцией запросов и ответов, используя заголовки сообщений.

Конфигурационный слой `KafkaConfig` отвечает за настройку компонентов Spring Kafka. Включает конфигурацию `ProducerFactory` для создания продюсеров Kafka, `ConcurrentMessageListenerContainer` для прослушивания топика с ответами, и бина `ReplyingKafkaTemplate`. Здесь же задаются параметры сериализации, десериализации и интеграции со Schema Registry.

Следующий подраздел посвящен разработке вычислительного сервиса на C++. Сервис спроектирован для асинхронной работы и параллельной обработки задач.

Для получения задач из Kafka реализован класс `KafkaListener`, который в бесконечном цикле опрашивает входной топик. При получении сообщения извлекаются полезная нагрузка и заголовки, установленные API-сервисом. Полезная нагрузка десериализуется из бинарного формата Avro в соответствующие структуры данных C++. Для этого были разработаны специализированные кодеки на основе шаблона `avro::codec_traits`.

После десериализации данные передаются в вычислительное ядро. Полученный результат сериализуется обратно в соответствующий Avro-формат. Перед сериализацией в начало байтового потока добавляется 5-байтовый префикс, содержащий идентификатор схемы из Schema Registry, что необходимо для корректной десериализации на стороне Java-сервиса.

Сформированное ответное сообщение отправляется в топик, указанный в заголовке исходного сообщения. В заголовки ответного сообщения копируется корреляционный идентификатор, что позволяет API-сервису сопоставить ответ с исходным запросом. Вся работа с Kafka в C++ компоненте ведется с помощью библиотеки modern-cpp-kafka.

Особое внимание уделено обработке ошибок. Перехватываются как исключения при взаимодействии с Kafka, так и ошибки, возникающие на этапе десериализации или самого вычисления. В случае ошибки в ответное сообщение вместо полезной нагрузки добавляется специальный заголовок с текстовым описанием проблемы. API-сервис, в свою очередь, проверяет наличие этого заголовка и, если он есть, возвращает клиенту HTTP-ответ со статусом ошибки и соответствующим сообщением.

В подразделе о межсервисной коммуникации обобщается весь процесс взаимодействия. Подчеркивается роль Avro-схем как строгого контракта данных между разнородными сервисами. Описывается весь жизненный цикл запро-

са: от отправки HTTP-запроса пользователем до получения им результата.

В последнем подразделе описан процесс сборки и тестирования приложения. С помощью Docker Compose разворачивается вся необходимая инфраструктура: Zookeeper, Kafka, Schema Registry, инструмент для мониторинга Kafka-UI, а также контейнеры с API-сервисом и вычислительным сервисом. Приведены примеры тестовых запросов, демонстрирующие корректную работу системы как в случае успешного расчета, так и при отправке некорректных данных или возникновении ошибок на стороне вычислителя.

В результате проделанной работы была создана серверная часть веб-приложения. Учтены такие особенности системы, как использования разных языков программирования для модулей, высокие затраты по времени на вычисления и потенциально долгий ответ пользователю.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была решена задача разработки и интеграции ключевых серверных компонентов для веб-приложения молекулярного моделирования Kvazar Online. Основное внимание было уделено созданию сервиса API на платформе Spring Boot и клиентской части вычислительного сервиса на языке C++, обеспечивающих взаимодействие через брокер сообщений Apache Kafka с использованием Apache Avro для сериализации данных и Confluent Schema Registry для управления схемами.

Проведен анализ предметной области молекулярного моделирования и определены требования к разрабатываемым компонентам. Были изучены и выбраны современные технологии, оптимально подходящие для построения распределенной, асинхронной и масштабируемой системы.

Спроектирована архитектура взаимодействия между API-сервисом и вычислительным сервисом. Разработаны Avro-схемы, определяющие четкий контракт данных для сообщений, передаваемых между сервисами, что обеспечило их корректную интерпретацию на обеих сторонах.

Таким образом, поставленные в дипломной работе цели были успешно достигнуты. Разработанные и интегрированные компоненты серверной части веб-приложения для молекулярного моделирования демонстрируют корректное функционирование и готовы к дальнейшему развитию и внедрению в полный цикл работы приложения.