Исследование эффективности использования мультиагентных моделей в современных микросервисных архитектурах

А. С. Бондаренко, Д.В. Королев, К.С. Зайцев

***Аннотация*. Целью настоящей работы является исследование возможности и способов применимости мультиагентных моделей в современных микросервисных архитектурах, так как сегодня программисты и архитекторы всё чаще отдают предпочтение именно такому подходу. Для достижения цели исследования в статье рассмотрены основные особенности разработки мультиагентных систем, и проведена оценка применимости и ожидаемого положительного эффекта при построении микросервисов с использованием агентов. В результате была разработана архитектура для внедрения готового медицинского программного комплекса по диагностике заболеваний щитовидной железы. Применение предложенных инструментов и архитектуры подтвердило правильность использования многоагентного подхода для построения микросервисов и продемонстрировали эффективность такого решения.**

***Ключевые слова* – микросервис, многоагентная система, мультиагентная система, микросервисная архитектура, облачные вычисления, программное обеспечение.**

I. ВВЕДЕНИЕ

Скоро распределенные автоматизированные системы с микросервисной архитектурой и управляющими системами на базе мультиагентов смогут выполнить функции, которые ранее считались невозможными для систем, созданных на традиционных архитектурных подходах. Мультиагентные технологии включают как разработку и использование мультиагентных систем, так и мультиагентное управление.

В последнее время много ученых решают задачи взаимодействия и управления в различных динамических системах. Часто помогают мультиагентные системы, которые могут решать задачи в различных областях, такие как распознавание в нейронных сетях, взаимодействие групп беспилотных летательных аппаратов, автоматизация систем управления, координация спутников и другие. Отдельно хочется выделить необходимость применения микросервисной архитектуры в задачах биомедицины, которые часто решаются масштабными автоматизируемыми системами, в которых взаимодействуют множества врачей разных специальностей, привлеченные эксперты, различные медицинские организации и иногда даже пациенты. Если функционал такой системы разделить на бизнес и платформенную части, и каждую в свою очередь разделить на более мелкие кусочки, выполняющие одну бизнес или техническую функцию, получится набор микросервисов, использующих разные технологии и взаимодействующие с множеством виртуальных или реальных акторов. В таком случае предложенная в статье архитектура будет лучшем решением и базисом построения такой системы.

Целью данной статьи является исследование эффективности использования мультиагентных моделей в современных микросервисных архитектурах программного обеспечения (ПО) и применение такой архитектуры для развертывания программы ассистента врача ультразвуковых исследований (УЗИ).

Для достижения поставленной цели необходимо решить следующие задачи:

1) Исследовать вопросы применения мультиагентных систем в интеллектуальных системах.

2) Проанализировать применение мультиагентных систем в микросервисных архитектурах.

3) Используя полученные результаты, построить архитектуру для решения реальной задачи из сферы биотехнологий на примере приложения для диагностики заболеваний щитовидной железы по результатам снимков ультразвукового исследования.

II. ОСОБЕННОСТИ РАЗРАБОТКИ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Ключевым понятием многоагентной системы является агент, который обладает набором сенсоров, помогающих ему взаимодействовать с другими агентами и внешней средой. Они имеют собственные источники мотивации для достижения поставленных

Статья получена 20.06.2023

Бондаренко Александр Сергеевич, Национальный исследовательский ядерный университет «МИФИ», аспирант, sasha-bond-95@mail.ru

Королев Денис Вячеславович, Национальный Исследовательский Ядерный Университет МИФИ, бакалавр, kdv024@ campus.mephi.ru

Зайцев Константин Сергеевич, Национальный исследовательский ядерный университет «МИФИ», профессор, kszajtsev@mephi.ru

перед ними целей. Если агенты имеют схожие цели, они группируются между собой. В свою очередь, группы агентов с общей целью объединяются в классы. Поскольку структура агентов близка к реальным структурам в мире, многоагентные системы широко применяются в различных сферах [1].

Все платформы разработки можно разделить на 3 большие группы:

• *Для моделирования*. Существуют различные платформы для моделирования, которые позволяют создавать 3D анимацию, моделировать экономические, социальные, биологические и другие системы. Они представляют собой закрытые платформы, в которых доступны инструменты для анализа взаимодействий между агентами и их наглядной визуализации [2,3].

• *Промышленные.* Этот класс систем обладает высокой степенью надежности, масштабируемости и интеграции с другими платформами. Хотя визуализация в ней отсутствует, она поддерживает стандарты взаимодействия и Web, а также сложные методы искусственного интеллекта.

• *Агенты виртуальных миров*, компьютерных игр и робототехники действуют в трехмерной среде, взаимодействуя с другими агентами и объектами. Обычно это сочетание первых двух типов платформ, которые представляют наиболее сложные среды, близкие к реальному миру.

Часть существующих международных стандартов для разработки МАС, являются неполными или устаревшими. Для разработки МАС можно воспользоваться двумя вариантами. В одном применяются агентные платформы, представляющие собой промежуточный уровень, находящийся между агентами и ОС. С другой стороны, существую специальные среды разработки, имеющие обширную область применения и функционал, пригодный для расширения и интеграции с другими системами.

Перечислим некоторые, наиболее востребованные свойства агента:

1. Цели, поставленные перед агентом, могут быть созданы в результате его интеллектуальной деятельности или заданы извне.

2. Агенты умеют воспринимать объекты окружающий среды, включая людей. Таким образом, им становится доступна некоторая модель внешнего мира. И чем сложнее эта модель, тем больше информации агент использует в своей работе.

3. В агенте должны быть заложены знания о его возможностях, условиях и последствиях его действий, чтобы он мог эффективно достигать свои цели и решать поставленные задачи.

Агенты искусственного интеллекта используют декларативное описание правил в виде "сущность-отношение" или "если-то" для определения своих возможностей, условий и последствий действий [4]. Машины вывода обрабатывают эти описания, чтобы подобрать последовательность действий, которая поможет достичь определенной цели. Они также могут проверять утверждения на основе имеющихся знаний.

Стоит отметить, что разработка агентных систем представляет собой сложную задачу, так как требует от разработчика не только высоких навыков в программировании, но и глубоких знаний в области искусственного интеллекта.

III. ПРИМИНЕНИЕ МУЛЬТИАГЕНТНЫХ МОДЕЛЕЙ В МИКРОСЕРВИСАХ

Рассмотрим некоторые примеры применения мультиагентных моделей в реализации проектов на основе микросервисной архитектуры.

В статье [5] описывается решение для интеллектуальных информационных систем, основанное на мультиагентном подходе и микросервисной архитектуре. Приводится пример использования этого решения при разработке автоматизированной системы интеллектуального поиска для тематической классификации различных источников информации в интернете. Комплекс выполняет поиск, классификацию и ранжирование веб-страниц на основе вероятности отнесения их к определенным категориям. Содержимое сайта включает различные компоненты: текст, изображения, аудио и т.д. В этом решении используется микросервисный подход, где основными элементами являются микросервисы - компоненты приложения с узконаправленной функциональностью, которые могут быть развернуты, и масштабированы автономно [6].

Выделяют два класса агентов: агенты-микросервисы и агенты-пользователи. Все упомянутые микросервисы относятся к типу агентов-микросервисов, которые связаны с определенными программными сущностями в системе. Важной характеристикой этого вида агентов является возможность масштабирования числа экземпляров при увеличении нагрузки.

В статье [7] строится архитектура, в которой брокер сообщений Apache Kafka является центральным элементом. Он применяется для связи между микросервисами, работающими в распределенной сети. Каждый микросервис подписывается на определенный топик сообщений, читает их, обрабатывает и отправляет в другой топик, который уже обрабатывается другим сервисом. Таким образом, каждый микросервис реагирует только на сообщения, относящиеся к его задаче, что упрощает взаимодействие между ними.

Каждый агент в обличии микросервиса представлен отдельным контейнером Docker, содержащим все необходимые библиотеки и зависимости для работы. Такой подход упрощает процесс развертывания, масштабирования и изменения системы. Для управления контейнерами сервисов используется портативная система контейнеризации с открытым исходным кодом Kubernetes, которая автоматизирует развертывание, масштабирование и операции приложений, а также позволяет разработчикам приложений использовать такие возможности, как самоконтроль, автоматизацию процессов, оркестровку хранилища и многое другое.

Сегодня контейнеризация приложений – не просто модный тренд, но и объективный подход для оптимизации процесса серверной разработки. Это достигается за счет унификации поддерживаемых инфраструктур (разработка, тестирование, опытная и промышленная эксплуатации), что в свою очередь значительно сокращает издержки на протяжении всего цикла жизни программного приложения.

В статье [8] обсуждаются различные подходы к созданию интеллектуальных методов и алгоритмов для управления рисками информационной безопасности в мультисервисных сетях. Авторы демонстрируют, что использование таких методов, необходимо для эффективного управления защищенной мультиагентной системы. В работе представлена математическая модель, которая была разработана и протестирована на оценку рисков информационной безопасности в системе.

В работе [9] рассматриваются системы с микросервисной архитектурой. Эти системы рассматриваются в качестве мультиагентных. Особое внимание уделено анализу возможных уязвимостей и угроз информационной безопасности рассматриваемых классических систем с микросервисной архитектурой. Автором разработана модель системы взаимодействия между агентами системы, в которой информация о разграничении доступа, аутентификационные данные микросервисов и идентификаторы агентов внешней среды хранятся на выделенном микросервисе. Этот микросервис, называемый в работе центральным узлом, взаимодействует с другими агентами посредством программного интерфейса приложения. Предложенная модель реализована как тестовый стенд, на котором собрана статистика, отражающая жизнеспособность системы. На основе полученных данных выявлены сильные и слабые стороны разработанной модели. Работа может быть полезна при планировании систем на основе микросервисной архитектуры с разграничением доступа к данным.

В статье [10] описывается модель микросервисной архитектуры, где используются слабо связанные, небольшие, заменяемые и автономные вычислительные микросервисы, взаимодействующие с помощью легковесных механизмов коммуникации.

В работе [11] рассказывается об облачной системе, которая предназначена для центров обработки данных, находящихся в разных географических местах. Разработка системы базируется на принципе мультиагентности и микросервисов, а для обеспечения взаимодействия между агентами используется разработанный специально для этой цели асинхронный протокол.

Алгоритм, специально разработанный для системы взаимодействия агентов, обеспечивает ее асинхронность. Реализация алгоритма выполнена на языке программирования Python и использует реляционные базы данных и системы очередей. Реляционная база данных служит для хранения запросов и ответов от агентов, а обмен YAML-сообщениями осуществляется с использованием идентификаторов.

В работе авторы исследовали применение RabbitMQ и PostgreSQL в кластерном режиме с шифрованием трафика, выявив особенности их использования. Разработанная модель может быть перспективной для работы с высоконагруженными распределенными системами

В работе [12] предлагается технология разработки интеллектуального мультиагентного решателя непроцедурных постановок вычислительных задач на распределенной модели предметной области. Используется агентный, ориентированный на прикладные микросервисы, способ организации вычислений на основе семантического взаимодействия прикладных агентов решателя.

Авторами приводится конечно-автоматная модель динамики функционирования агентов решателя. Предложенная технология демонстрируется на примере построения распределенного решателя для исследования поведения траекторий автономных двоичных динамических систем.

В работе [13], авторы обсуждают инструменты управления вычислениями в пакете прикладных микросервисов, который предназначен для использования при решении задач качественного исследования двоичных динамических систем на основе метода булевых ограничений. Вычислительная модель предметной области и архитектура агента обеспечивают возможность проведения многовариантных, параллельных и конвейерно-параллельных вычислений на выделенных ресурсах. Предложенный подход демонстрируется на примере анализа структуры пространства состояний сдвигового регистра.

Проанализировав представленные выше публикации и возможности применения мультиагентных моделей при построении архитектуры микросервисов, становится очевидно, что такие модели помогут ускорить процесс разворачивания больших программных комплексов для использования в различных средах разработки и тестирования. Кроме того, благодаря своей природе, агенты представленные микросервисами облегчают процесс управления и оркестрации кластером конейтенеризированных приложений. Далее, перед авторами была поставлена задача адаптировать существующий программный комплекс по диагностике заболеваний щитовидной железы в микросервисную архитектуру для упрощения процесса внедрения и управления комплексом. Ожидается что такой подход существенно ускорит время установки на различные тестовые и промышленные стенды и снизит порог входа для администраторов системы при эксплуатации и сопровождении комплекса.

IV. ПОСТРОЕНИЕ АРХИТЕКТУРЫ ДЛЯ КОМПЛЕКСА ДИАГНОСТИКИ ЗАБОЛЕВАНИЙ ЩИТОВИДНОЙ ЖЕЛЕЗЫ ПО СНИМКАМ УЗИ

В настоящей работе мы рассматриваем процесс создания архитектуры программного приложения, которое поможет врачам диагностировать

заболевания щитовидной железы на основе анализа снимков УЗИ. Для врачей диагностика снимков УЗИ является сложной задачей, но существует система классификации новообразований (узлов) щитовидной железы – TI-RADS [14], которая позволяет определить опасность новообразования, его границы и др. характеристики. Мы совместно с НМИЦ Эндокринологии разработали интеллектуальную систему ассистента врача УЗИ диагностики, которая оказывает содействие в определение класса TI-RADS и облегчает работу врача, подсказывая ему свои решения.

Созданная система является веб-приложением клиент-серверного типа, которое позволяет врачам загружать УЗИ кинопетли в различных форматах и с помощью нейронных сетей получать границы узлового образования и его класс по TI-RADS. Программное приложение может быть установлено в отдельных клиниках или на платформах для всеобщего использования. Собирая информацию от каждого врача-диагноста, приложение может использоваться для дообучения профильных нейронных сетей. В связи с большим количеством матричных умножений, вызванных использованием нейронных сетей, разработанное веб-приложение поддерживает работу с графическими процессорами (GPU) для уменьшения времени обработки УЗИ снимков.

Бизнес-единица приложения представляет собой веб-приложение, написанное на языке Python, с использованием фреймворков Jango, FastAPI, PyTorch, Celery.

Разработанный комплекс представляет собой набор сильно связанных между собой программных компонентов, запускаемых в одном слое инфраструктуры на локальном компьютере или удаленном сервере. В таком виде запустить различные экземпляры, например, при желании установить комплекс и в тестовой, и в промышленной средах или, исходя из нагрузки на вычисления, управлять количеством экземпляров, работа трудная. В данном исследовании мы постарались разбить существующие модули по бизнес-целям, где каждая бизнес-цель является целью агента, а многоагентный подход применен для построения микросервисной архитектуры, в которой будет запущен комплекс диагностики заболеваний.

Используя знания, полученные при анализе публикаций, представленных в предыдущем разделе, был выбран следующий технологический стек. Для связи между Django-сервером и сервером с моделями нейронных сетей используется решение Redis в качестве брокера сообщений. PostgreSQL выбрана в качестве СУБД благодаря поддержке множества современных инструментов для разработки сервисов и ее способности эффективно работать с большим объемом данных, так как она основана на объектно-реляционной модели. Для оркестрации контейнеров, был выбран Kubernetes, таким образом, построенная архитектура будет представлена через его сущности.

Рассмотрим наиболее общий вариант, когда все инфраструктурные компоненты развернуты в контейнеризируемой среде, включая такие нетрадиционные для нее, как брокеры сообщений и СУБД, которые могут быть вынесены за пределы этой архитектуры при использовании в промышленных масштабах, поскольку ввиду особенностей реализации и развертывания не всегда удобно представимы в виде микросервисов. Пример архитектуры представлен на рисунке 1.

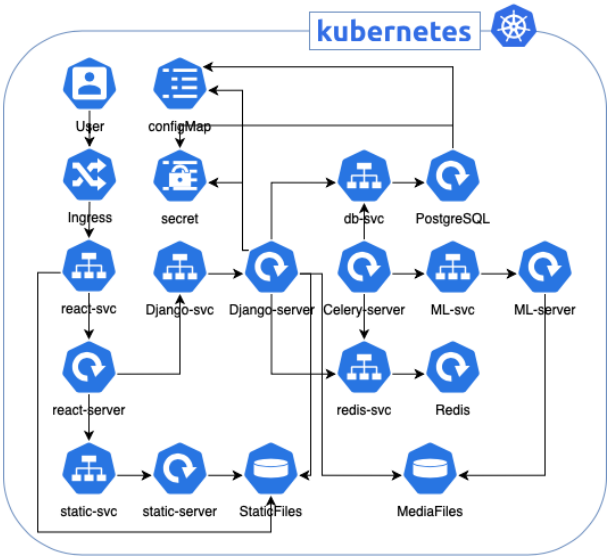

Рисунок 1 – Kubernetes сущности архитектуры автономного приложения

Сущности Kubernetes в данном случае представлены семью деплойментами (Deployment - объект Kubernetes, представляющий работающее приложение в кластере). Четыре из них, поднимают поды (Pod - минимальная вычислительная единица, которую можно запустить в кластере Kubernetes) с исполняемым Python кодом (Django, Static, Celery и ML Service), один под обслуживает JS React приложение, оставшиеся два - представляют собой вышеобозначенные PostrgreSQL и Redis сервер. Сущность Deployment описывает, как будут обновляться поды и реплика-сеты (ReplicaSets). Мы описываем желаемое состояние развертывания в деплойменте, а контроллер развертывания изменяет фактическое состояние на желаемое согласно заданному поведению.

Существуют способы заставить деплоймент создавать новые реплика-сеты или заменять существующий деплоймент, адаптировав все новые изменения в нем, мы выбрали первый вариант. В выше показанной схеме каждый из семи модулей комплекса представлен в виде деплоймента, что сделано для упрощения. Строго говоря, сам по себе деплоймент лишь описывает каким образом будет

происходить развертывание модуля и какое поведение будет при успешном / неуспешном развертывании или изменения количества реплик. Сам модуль (его исполняемая программа / основной процесс / код) поднимается в контейнере, именуемым подом. Количество и статус подов контролируется ReplicaSet, которым так же управляет Deployment. Более подробная схема вокруг одного модуля (например, Django-server) представлена на рисунке 2.

Множество подов позволяют существенно повысить отказоустойчивость всей системы, так как таким образом мы реплицируем бизнес модуль. В случае если одна из реплик станет недоступной, механизм service discovery узнает об этом, и выведет его из топологии сети. Кроме того, запуск нескольких подов одного модуля означают его параллельную работу, тем самым еще повышая и скорость ответа для пользователя или вызываемой службы, так как в этом случае запрос идет не на один экземпляр сервиса и становится в очередь, а равномерно согласно round-robin уходит на различные экземпляры (поды).

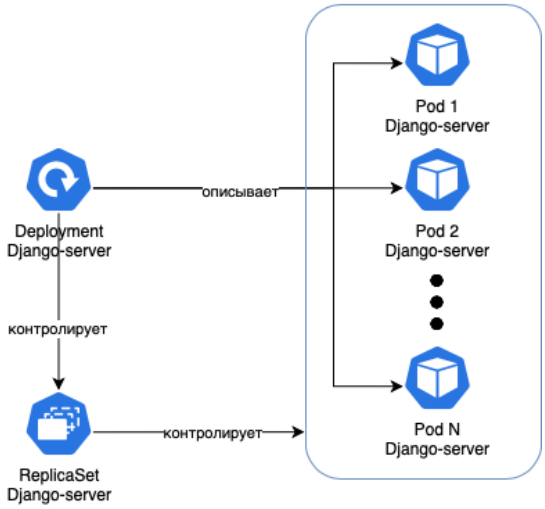

Рисунок 2 – Структура связи между деплойментами и подами Django-server

Каждый деплоймент в данном случае – микросервис, который имеет Kubernetes-сервис для доступности в топологии сети внутри контейнеров. Сервис в этом контексте — это метод предоставления доступа к сетевому приложению, которое работает как один или несколько модулей в кластере. Основная их цель заключается в том, что нет необходимости изменять существующее приложение для использования стороннего механизма обнаружения служб. Можно запускать код в подах, будь то код, предназначенный для облачного мира, или более старое приложение, которое было контейнеризировано. Мы используем сервисы, чтобы сделать этот набор модулей доступным в сети и клиенты могли взаимодействовать с ним.

Хранилища MediaFiles и StaticFiles используются для сохранения статических файлов отдаваемые веб сервером и пользовательских рентген / УЗИ изображений, загружаемых в систему, для последующей обработки ML сервером. Deployment описывает какие хранилища примонтировать к какому поду, а уже сами хранилища непосредственно монтируются в файловую систему модуля.

Компонент configMap необходим для сохранения переменных окружения всей системы и конфигурационных файлов, позволяющих изменять конфигурацию сервиса на лету. Компонент secret нечто схожее с configMap, но содержит в себе чувствительные данные, такие как пароли подключения к БД, сертификаты безопасности и так далее. Основная разница между secrets и configMaps заключается в том, что данные в secrets закодированы с помощью Base64. Мы рекомендуем использовать secrets для конфиденциальных данных и configMaps для не конфиденциальных данных (для чего это в данной статье). Если рассмотреть подробно взаимодействие описанных выше сущностей вокруг уже рассмотренного модуля, мы увидим решение, представленное на рисунке 3.

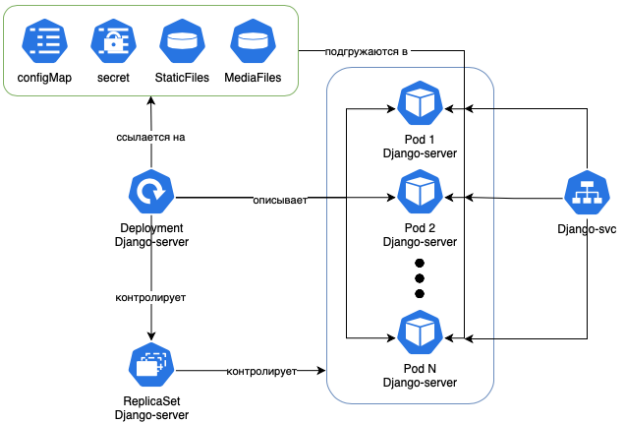

Рисунок 3 – Структура связей различных сущностей с модулем Django-server

Для захода внешнего пользователя в кластер и попадания на главную страницу сервиса необходим Ingress компонент, обеспечивающий видимость приложения в сети предприятия или интернете.

Важно отметить, что построенную архитектуру легко расширить на случай развёртывания комплекса в существующей платформе, где в промышленном контуре предоставляются для использования уже готовые системы мониторинга, СУБД, компоненты логгирования и так далее. В нашем случае, такие элементы как Redis и PostgreSQL, которые в виду особенностей работы и реализации развёртывания не соответствуют парадигме микросервисов и должны быть вынесены за пределы этой архитектуры. У нас нет необходимости перестраивать архитектуру

целиком, нужно только внести минимальные изменения. Положим, PostgreSQL и Redis развернуты отдельно (в рамках используемой платформы или подняты самостоятельно как внешние отдельные сервисы (инстансы) и у нас есть доступ к этим системам, реквизиты подключения и т.д. Тогда необходимо убрать два одноименных деплоймента, и перенаправить ассоциированные Kubernetes-сервисы вместо деплойментов на новую сущность endpoints. Сущность помогает определить каким именам в сети соответствуют какие IP адреса, поскольку в этом случае Kubernetes сам ничего не знает о внешних системах. По умолчанию сервис имеет тип clusterIP что означает, что он может определять имена хостов согласно внутренней таблице маршрутизации Kubernetes, но для использования внешних сервисов необходимо поменять тип, это может быть NodePort, LoadBalancer или ExternalName в зависимости от потребностей и возможностей. Новая архитектура будет иметь вид, изображённый на рисунке 4.

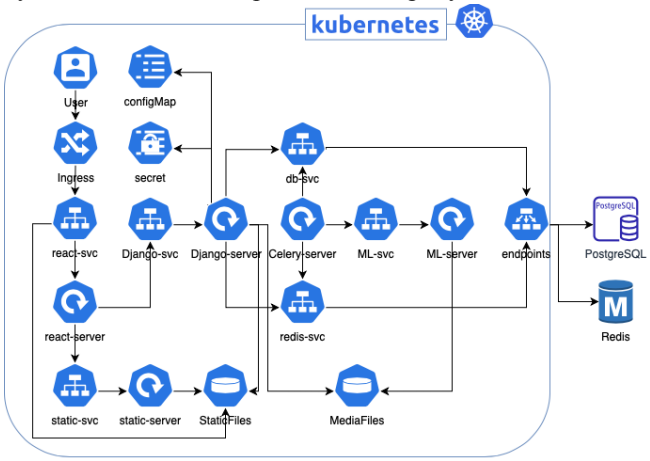

Рисунок 4 – Kubernetes сущности архитектуры платформенного приложения

Каждый под представляет собой автономного агента, объединенных в группы агентов, которые в свою очередь представляют отдельный бизнес модуль комплекса. Взаимодействующие между собой группы агентов внутри кластера посредством обмена сообщениями решают общую задачу диагностики заболеваний щитовидной железы по снимкам УЗИ. Таким образом, построенная микросервисная архитектура представляет собой многоагентную систему.

V. ДИСКУССИЯ ПО ТЕМЕ ИССЛЕДОВАНИЙ

На сегодняшний момент тема использования многоагентных моделей при создании микросервисов широко обсуждается научным сообществом и прикладными инженерами по всему миру. В статье авторы широко освещают использование таких моделей совместно с процессом контейнеризации приложений и подчеркивают важность использования подхода при построении систем для облачных вычислений.

Ученое сообщество активно исследует применимость мультиагентных моделей в различных отраслях. Так, например в статье [15] из области нейропедиатрии, авторы рассказывают о преимуществе такого подхода при создании системы по детектированию и предсказанию эпилепсии, чьи задачи (с точки зрения инженерии) довольно схожи с комплексом по выявлению патологий щитовидной железы, исследуемым в рамках нашей статьи. В статье [16] показывают применение многоагентной системы для построения генеративного видения будущих патогентов вируса COVID-19. Кроме сферы биомедицины, такие модели активно применяются в других отраслях, например из публикаций [17, 18] можно узнать о том, как используются мультиагенты для обучения с подкреплением системы управления маршрутами и сфетофорами для автомобилей экстренных служб.

На текущий момент все описанные выше подходы и технологии активно развиваются, как сообществом, так и вендорами, поэтому в ближайшее время количество исследований в данной тематике будет увеличиваться, изменяя и дополняя существующие факты, расширяя области применимости многоагентных моделей в мироксервисных архитектурах.

VI. ЗАКЛЮЧЕНИЕ

В мультиагентных технологиях решение задачи достигается благодаря самоорганизации множества программных агентов, которые конкурируют, кооперируются между собой, и имеют свои критерии, предпочтения и ограничения. В отличие от классического подхода, который основывается на детерминированном алгоритме комбинаторного поиска вариантов решения, в мультиагентных технологиях решение считается найденным, когда агенты достигают неулучшаемого консенсуса в ходе своих недетерминированных взаимодействий, что является временным равновесием или балансом интересов.

В рамках исследуемой в статье темы была исследована применимость мультиагентных систем в интеллектуальных системах и системах с микросервисной архитектурой. Используя полученные знания о мультиагентах, и совместив их с микросервисным подходом, была построена архитектура приложения для диагностики заболеваний щитовидной железы по снимкам УЗИ. Микросервисные агенты представлены в виде отдельных контейнеров Docker, обеспечивающих их изоляцию. Каждый контейнер содержит все необходимые зависимости и библиотеки для работы агента, который связан с определенной программной сущностью в системе. Значительной особенностью этих агентов является возможность масштабирования путем запуска необходимого количества экземпляров при увеличении нагрузки, что осуществляется

благодаря созданию и тонкой настройке различных сущностей внутри Kubernetes среды.

Созданная архитектура позволяет минимальными усилиями внедрить комплекс в существующие платформы, заменяя внутренние компоненты платформенными или вынести некоторые компоненты, такие как базы данных и брокеры сообщений за пределы этой архитектуры. Более того, возможно одновременно использовать как полностью автономно работающую версию (например, для цели разработки или тестирования), так и платформенную, для использования в промышленной эксплуатации. Такой подход позволяет существенно сократить время внедрения комплекса и снижает сложность сопровождения системы, при условии, что персонал сопровождения обучен управлению микросервисами. Полученная архитектура помогает легко масштабировать мощности приложения в зависимости от потребностей медицинской организации, полноты и объёма медицинских исследований.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

1. Тарасов, В. Б. (1998). Агенты, многоагентные системы, виртуальные сообщества: стратегическое направление в информатике и искусственном интеллекте. Новости искусственного интеллекта, (2), 5-63.
2. Li Y. et al. Learning distilled collaboration graph for multi-agent perception //Advances in Neural Information Processing Systems. – 2021. – Т. 34. – С. 29541-29552.
3. Plappert S., Gembarski P. C., Lachmayer R. Multi-agent systems in mechanical engineering: a review //Agents and Multi-Agent Systems: Technologies and Applications 2021: Proceedings of 15th KES International Conference, KES-AMSTA 2021, June 2021. – Singapore : Springer Singapore, 2021. – С. 193-203.
4. Рассел, С. (2006). Искусственный интеллект. Современный подход.
5. Агеев С.А., Саенко И.Б. Управление рисками информационной безопасности защищенной мультисервисной сети специального назначения на основе интеллектуальных мультиагентов. // Вестник Югорского государственного университета, 2020 г. Выпуск 3 (58). С. 47–52
6. Граса Г. Архитектура микросервисов. [Электронный ресурс]: https://habr.com/ru/company/vk/blog/320962/ (Дата обращения: 10.02.2023).
7. Радченко Г.И. Сервис-ориентированная архитектура [Электронный ресурс]: https://glebradchenko.susu.ru/courses/bachelor/odp/2013/SUSU_Distr_08_SOA_02.pdf (Дата обращения: 25.01.2023).
8. Бурлуцкий, В. В., Керамов, Н. Д., Балуев, В. А., Изерт, М. И., & Якимчук, А. В. (2020). РАЗРАБОТКА МУЛЬТИАГЕНТНОЙ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ КЛАССИФИКАЦИИ И РАНЖИРОВАНИЯ МАТЕРИАЛОВ В СЕТИ ИНТЕРНЕТ. Вестник Югорского государственного университета, (3 (58)), 47-52.
9. Меренков, Д. Н. (2021). ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ МОДЕЛИ СИСТЕМЫ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ. Инновации. Наука. Образование, (26), 1658-1671.
10. Самохин, Н. Ю., Орешкин, А. А., & Супрун, А. С. (2019). Реализация протокола обмена данными между программными агентами в облачной инфраструктуре в географически распределенных центрах обработки данных. Научно-технический вестник информационных технологий, механики и оптики, 19(6), 1086-1093.
11. Богданова, В. Г., & Пашинин, А. А. (2018). Разработка самоорганизующейся мультиагентной системы децентрализованного управления распределенным решением прикладных задач. Информационные и математические технологии в науке и управлении, (3 (11)), 115-126.
12. Опарин, Г. А., Богданова, В. Г., & Пашинин, А. А. (2019). Управление конвейерно-параллельными вычислениями при решении задач качественного исследования двоичных динамических систем на основе метода булевых ограничений. Информационные и математические технологии в науке и управлении, (3 (15)), 79-90.
13. Городецкий, В. И., Грушинский, М. С., & Хабалов, А. В. (2015). Многоагентные системы: обзор современного состояния теории и практики. при поддержке Российского фонда фундаментальных исследований (грант, (96-01).
14. Botz B. European Thyroid Association TIRADS, 2021 [Электронный ресурс]: https://radiopaedia.org/articles/european-thyroid-association-tirads (Дата обращения: 17.12.2022).
15. Bertoncelli C. M. et al. PredictMed-epilepsy: A multi-agent based system for epilepsy detection and prediction in neuropediatrics //Computer Methods and Programs in Biomedicine. – 2023. – Т. 236. – С. 107548.
16. Lee B. et al. Generative design for COVID-19 and future pathogens using stochastic multi-agent simulation //Sustainable Cities and Society. – 2023. – С. 104661.
17. Su H. et al. EMVLight: A multi-agent reinforcement learning framework for an emergency vehicle decentralized routing and traffic signal control system //Transportation Research Part C: Emerging Technologies. – 2023. – Т. 146. – С. 103955.
18. Chu T. et al. Multi-agent deep reinforcement learning for large-scale traffic signal control //IEEE Transactions on Intelligent Transportation Systems. – 2019. – Т. 21. – №. 3. – С. 1086-1095.

Study the efficiency of using multi-agent models in modern microservice architectures

A.S. Bondarenko, D.V. Korolev, K.S. Zaytsev

***Abstract* – The purpose of this work is to explore the possibility and ways of applicability of multi-agent models in modern microservice architectures, since today programmers and architects increasingly prefer this approach. To achieve the goal of the study, the article considers the main features of the development of multi-agent systems, and assesses the applicability and expected positive effect when building microservices using agents. As a result, an architecture was developed for the implementation of a ready-made complex for diagnosing thyroid diseases. The results of applying the proposed tools and architecture confirmed the correctness of using a multi-agent approach to build microservices and demonstrated the effectiveness of such a solution.**

***Keywords* – microservice, multi-agent system, microservice architecture, cloud computing, software.**

REFERENCES

1. Tarasov, V. B. (1998). Agents, multi-agent systems, virtual communities: a strategic direction in computer science and artificial intelligence. Artificial Intelligence News, (2), 5-63.
2. Li Y. et al. Learning distilled collaboration graph for multi-agent perception //Advances in Neural Information Processing Systems. – 2021. – vol. 34. – pp. 29541-29552.
3. Plappert S., Gembarski P. C., Lachmayer R. Multi-agent systems in mechanical engineering: a review //Agents and Multi-Agent Systems: Technologies and Applications 2021: Proceedings of 15th KES International Conference, KES-AMSTA 2021, June 2021. – Singapore : Springer Singapore, 2021. – pp. 193-203.
4. Russell, S. (2006). Artificial intelligence. Modern approach.
5. Ageev S.A., Saenko I.B. Management of information security risks of a secure multi-service special-purpose network based on intelligent multi-agents. // Bulletin of the Yugra State University, 2020 Issue 3 (58). pp. 47–52
6. Graça G. Microservices Architecture. [Electronic resource]: https://habr.com/ru/company/vk/blog/320962/ (Date of access: 02/10/2023).
7. Radchenko G.I. Service-Oriented Architecture [Electronic resource]: https://glebradchenko.susu.ru/courses/bachelor/odp/2013/SUSU_Distr_08_SOA_02.pdf (Date of access: 01/25/2023).
8. Burlutsky, V. V., Keramov, N. D., Baluev, V. A., Izert, M. I., & Yakimchuk, A. V. (2020). DEVELOPMENT OF A MULTI-AGENT INTELLIGENT SYSTEM FOR SOLVING THE PROBLEMS OF CLASSIFICATION AND RANKING OF MATERIALS ON THE INTERNET. Bulletin of the Yugra State University, (3 (58)), 47-52.
9. Merenkov, D. N. (2021). ENSURING INFORMATION SECURITY OF A SYSTEM MODEL WITH MICROSERVICE ARCHITECTURE. Innovation. The science. Education, (26), 1658-1671.
10. Samokhin, N. Yu., Oreshkin, A. A., & Suprun, A. S. (2019). Implementation of a data exchange protocol between software agents in a cloud infrastructure in geographically distributed data processing centers. Scientific and technical bulletin of information technologies, mechanics and optics, 19(6), 1086-1093.
11. Bogdanova, V. G., & Pashinin, A. A. (2018). Development of a self-organizing multi-agent system for decentralized control of a distributed solution of applied problems. Information and Mathematical Technologies in Science and Management, (3 (11)), 115-126.
12. Oparin, G. A., Bogdanova, V. G., & Pashinin, A. A. (2019). Control of pipeline-parallel computing in solving problems of qualitative research of binary dynamical systems based on the Boolean constraint method. Information and mathematical technologies in science and management, (3 (15)), 79-90.
13. Gorodetsky, V. I., Grushinsky, M. S., & Khabalov, A. V. (2015). Multi-agent systems: a review of the current state of theory and practice. with the support of the Russian Foundation for Basic Research (grant, (96-01).
14. Botz B. European Thyroid Association TIRADS, 2021 [Electronic resource]: https://radiopaedia.org/articles/european-thyroid-association-tirads (Date of access: 12/17/2022).
15. Bertoncelli C. M. et al. PredictMed-epilepsy: A multi-agent based system for epilepsy detection and prediction in neuropediatrics //Computer Methods and Programs in Biomedicine. – 2023. – Vol. 236. – С. 107548.
16. Lee B. et al. Generative design for COVID-19 and future pathogens using stochastic multi-agent simulation //Sustainable Cities and Society. – 2023. – С. 104661.
17. Su H. et al. EMVLight: A multi-agent reinforcement learning framework for an emergency vehicle decentralized routing and traffic signal control system //Transportation Research Part C: Emerging Technologies. – 2023. – Vol. 146. – С. 103-955.
18. Chu T. et al. Multi-agent deep reinforcement learning for large-scale traffic signal control //IEEE Transactions on Intelligent Transportation Systems. – 2019. – Vol. 21. – Issue. 3. – pp. 1086-1095.