DOI: 10.18137/RNU.V9187.23.01.P.37 УДК 004.421+004.4’2

В.Н. Клоков, С.Е. Вечерская

ЗАДАЧИ И ЭВОЛЮЦИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Аннотация. В обзоре рассмотрена история развития микросервисной архитектуры, определены понятия микросервисной архитектуры и микросервиса, выполнено сравнение монолитной и микросервисной архитектур, что позволило обосновать выбор преимущественной архитектуры. Также рассмотрено несколько успешных кейсов использования микросервисной архитектуры и описаны условия, при которых выбор микросервисов является оптимальным.

Ключевые слова: системная архитектура, микросервисы, монолит, системы управления, сервисно ориентированная архитектура.

V.N. Klokov, S.E. Vecherskaya

TASKS AND EVOLUTION OF MICROSERVICE ARCHITECTURE

Abstract. The review examines the history of the development of micro-service architecture, defines the concepts of micro-service architecture and microservice, compares monolithic and microservice architectures, which made it possible to justify the choice of the preferred architecture. Several successful cases of using micro-service architecture are also considered and the conditions under which the choice of microservices is optimal are described.

Keywords: system architecture, microservices, monolith, control systems, service-oriented architecture.

Введение

Одним из ведущих современных трендов развития бизнес-структур является их укрупнение с углублением диверсификации. Соответствующие этому изменения в построении информационных систем, в том числе систем управления, должны обеспечивать как целостность управления процессами, так и максимально возможную эффективность с учетом усложнения архитектуры. Одним из вариантов решения этой задачи является выбор модульных структур с соответствующим программным обеспечением. Ключевым является создание систем управления на основе набора независимых, но взаимодействующих друг с другом модулей, в котором каждый модуль отвечает за решение определенного блока задач и может быть видоизменен в соответствии с потребностями данного блока.

В данной работе рассматривается понятие «микросервисная архитектура», дан обзор истории ее развития, а также проведено сравнение с другими популярными архитектурами. В рамках исследования была поставлена задача – определить, является ли микросервисная архитектура абсолютно необходимой в современном мире, или это лишь один из трендов. Заметим, что в настоящее время существует крайне мало теоретических исследований в этом направлении, поэтому задача представляется актуальной и своевременной.

Краткая история создания микросервисов

Микросервисная архитектура появилась в результате эволюции сервисно ориентированной архитектуры (SOA). Впервые над идеей микросервисов начал работать Питер Роджерс, ИТ-архитектор компании NetKernel, в 1999 году. Целью его исследования было сделать код менее хрупким, более гибким и устойчивым к большим изменениям.

Клоков Владимир Николаевич
магистрант кафедры информационных систем в экономике и управлении Института информационных систем и инженерно-компьютерных технологий, Российский новый университет, Москва. Сфера научных интересов: цправление проектами; оптимизация процессов. Электронный адрес: clokov2015@yandex.ru

Вечерская Светлана Евгеньевна
кандидат химических наук, доцент, доцент кафедры информационных систем в экономике и управлении, Российский новый университет, Москва. Сфера научных интересов: эффективность управления; эконометрика. Автор более 40 опубликованных научных работ. ORCID: 0000-0001-6721-1388; SPIN-код: 1343-7927.
Электронный адрес: s.vecherskaya@bk.ru

Первые результаты исследования были представлены им на конференции Web Services Edge в 2005 году. Он выдвинул идею, что необходимо не просто создавать отдельные сервисы по SOA – самой популярной архитектуре веб-разработки на тот момент, но распределять данные по конкретным микросервисам и передавать их с помощью коннекторов. В своей презентации Роджерс наглядно пояснил, как распределение данных между отдельными сервисами позволяет гибко подстраиваться под любые изменения, а также увеличить простоту кода в отдельно взятых сервисах.

Параллельно с ним Юваль Леви, архитектор программного обеспечения и руководитель компании IDesign, в 2007 году высказал свою идею, весьма схожую с понятием «микросервис», он призвал к созданию систем, где каждый отдельный класс является сервисом. Это требует использования технологий, которые могут поддерживать такое детальное использование сервисов, и Леви самостоятельно расширил Windows Communication Foundation (WCF), взяв каждый класс и рассмотрев его как сервис, сохраняя при этом традиционную модель программирования классов.

Также в 2005 году Алистер Кокберн, американский ученый-компьютерщик, написал о гексагональной архитектуре, которая представляет собой шаблон проектирования программного обеспечения, используемый наряду с микросервисами. Этот шаблон делает возможным проектирование микросервиса, поскольку он многоуровнево изолирует бизнес-логику от вспомогательных служб, необходимых для развертывания и запуска микросервиса, полностью независимого от других [1]. Широкое применение идеи микросервисов началось в 2011 году, когда на различных конференциях (например, "A Workshop of Software Architects", май 2011 года) всё чаще стали использовать термин «микросервис». С 2012 по 2014 годы о глобальном внедрении микросервисов в архитектуру своих сервисов сообщили такие компании, как Amazon, Netflix и Twitter. По данным компании O'Reilly, на 2020 год микросервисы использовались в 77 % компаний мира, и таких компаний становилось всё больше [2].

Определение и основные понятия микросервисов

Еще недавно программные приложения создавались преимущественно на базе монолитной архитектуры, где каждое приложение представляет собой самостоятельную единицу. Такой подход приносил пользу многим командам разработчиков до тех пор, пока

приложения не становились слишком сложными, и чтобы изменить небольшой участок кода в монолитной системе, приходилось заново собирать всю систему, тестировать ее и развертывать новую версию приложения.

Затем появились микросервисы. Программные системы стали разбиваться на небольшие элементы, которые можно разрабатывать и развертывать независимо друг от друга. Микросервисная архитектура развивалась при поддержке приверженцев DevOps, которым требовалась быстрая поставка обновлений – новых возможностей, исправлений багов и улучшений безопасности. Кроме того, с такой архитектурой многие компании могли переписать устаревшие приложения с использованием современных языков программирования и обновленного стека технологий [3].

Микросервисная архитектура, или микросервисы, – это подход к созданию приложения в виде набора независимо развертываемых сервисов, которые являются децентрализованными и разрабатываются независимо друг от друга, слабо связаны и легко обслуживаются. Монолитное приложение создается как единое и неделимое целое, тогда как в микросервисной архитектуре его разбивают на множество независимых модулей, каждый из которых вносит свой вклад в общее дело.

Микросервисы неразрывно связаны с DevOps, поскольку лежат в основе методики непрерывной поставки, благодаря которой команды могут быстро адаптироваться к требованиям пользователей. Микросервис – это веб-сервис, отвечающий за один элемент логики в определенной предметной области. Приложения создаются как комбинации микросервисов, каждый из которых предоставляет функциональные возможности в своей предметной области. Микросервисы взаимодействуют друг с другом через API-интерфейсы, такие как REST или gRPC, но не обладают информацией о внутреннем устройстве других сервисов.

На Рисунке 1 представлен пример использования микросервисной архитектуры [3].

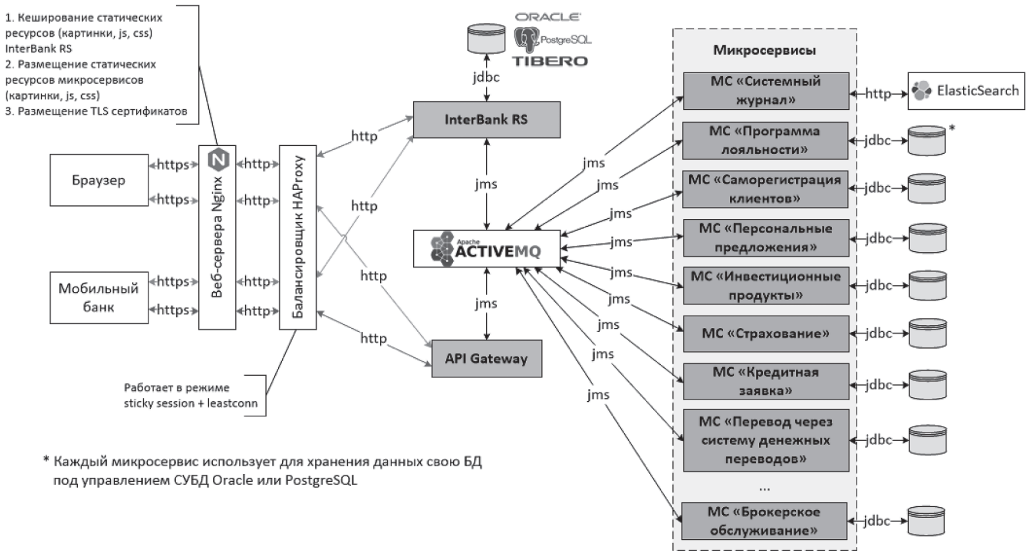

Рисунок 1. Пример использования микросервисов
Источник: Микросервисная архитектура от А до Я [6].

Микросервисы как противопоставление монолиту

Ни одна архитектура не известна так хорошо, как проверенное и настоящее монолитное приложение. Сторона клиента, сторона сервера и база данных соединяются в один JAR-файл и отправляются на сервер для развертывания. Его легко разрабатывать, тестировать и развертывать. Однако при попытке сделать сервис более быстрым и эффективным у монолитной архитектуры выявляется много недостатков. Решением проблемы может быть переход на микросервисы. Микросервисная архитектура является альтернативой классическому монолиту и решает многие проблемы, связанные с последним. Основная идея микросервисов состоит в том, чтобы разбить бизнес-логику и уровни доступа к данным на независимые сервисы. Каждая служба имеет собственную базу данных и ориентирована на определенные бизнес-возможности. Микросервис может взаимодействовать как с клиентской стороной, так и с другими службами через четко определенный API [4]. В Таблице представлен свод плюсов и минусов монолитов и микросервисов.

Таблица

Сравнение монолитной и микросервисной архитектуры*

Оценка	Монолит	Микросервисы
Плюсы	• проще в реализации • легко разворачивается • легко масштабируется	• можно обновлять только необходимые модули сервиса • задачи можно разбивать между небольшими командами • независимость сервисов друг от друга
Минусы	• невозможно вносить доработки в код, не согласовав их со всей командой • при минимальной поломке может сломаться весь сервис • ухудшение производительности при слишком большом приложении	• сложность понимания архитектуры при большом количестве коннекторов • раздельные базы данных • не вполне оправданный тренд; иногда в микросервисах нет необходимости, но они всё равно используются

**Источник: Монолитная vs Микросервисная архитектура [7].*

На Рисунке 2 представлено наглядное сравнение монолитной и микросервисной архитектур.

Кейсы применения микросервисной архитектуры

Для понимания, в каких ситуациях лучше использовать микросервисы, следует рассмотреть несколько кейсов, где может быть использована микросервисная архитектура [5].

Приложения, работающие с большими данными. AL/ML – один из лучших вариантов использования архитектуры микросервисов. Приложения для работы с большими данными требуют сложной архитектуры, ориентированной на конвейер данных, где каждый этап управляет одной (или несколькими) конкретными задачами, например, сбором, обработкой, доставкой, хранением и др. Такие слабо связанные блоки отлично подходят для микросервисов.

Приложения для обработки данных в реальном времени. Наиболее распространенными вариантами использования являются потоковые платформы, такие как YouTube или SoundCloud, приложения электронного банкинга, системы управления дорожным движением, онлайн-бронирование и др., – приложения, выполняющие операции в режиме реального времени для быстрого получения результата.

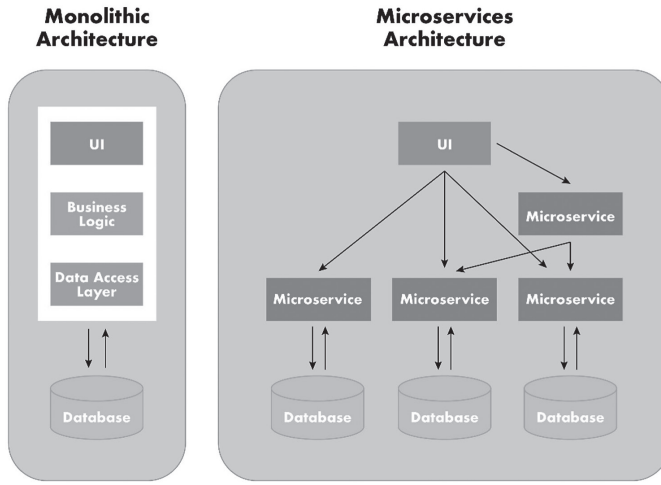

Рисунок 2. Сравнение монолита и микросервисов
Источник: Монолитная vs Микросервисная архитектура [7].

Крупномасштабные системы со сложной логикой. Стремительно растущие организации (например, социальные сети, средства массовой информации, e-commerce) требуют выдающейся гибкости и масштабируемости, которые в точности обеспечивает архитектура микросервисов. Это помогает компаниям быстро находить узкие места или задействовать новые передовые технологии. Более того, возможность частого развертывания на уровне взаимодействия обеспечивает еще более быструю масштабируемость [8].

Заключение

Очевидно, что микросервисная архитектура имеет как свои плюсы, так и свои минусы, и не всегда использование именно данной архитектуры является лучшим решением. Свои наилучшие качества микросервисы проявляют в следующих случаях:

- при работе с большим количеством данных, которые необходимо пересылать и обрабатывать между большим количеством сервисом;
- в больших компаниях, поскольку это позволяет распределить микросервисы между небольшими командами, которые будут минимально зависимы друг от друга;
- при увеличении производительности сервиса. Если функциональность сервиса выросла до таких масштабов, что ее производительность начала ухудшаться, то необходимо задуматься над переходом к микросервисной архитектуре.

Для микросервисов не существует строго регламентированных правил, в каких случаях можно использовать данную архитектуру, а в каких – нет. Они могут применяться как для маленького стартапа, так и в больших компаниях. Всё зависит от конкретных целей сервиса, и данный вопрос должен решаться в команде разработки на уровне стратегического и операционного менеджмента, поскольку дальнейшее развитие сервисов и всей ИТ-инфраструктуры в целом напрямую зависит от архитектуры и направления развития, который был выбран.

Литература

1. *Опарин Г.А., Богданова В.Г., Пашинин А.А.* Микросервисы как фундаментальная основа распределенного сборочного программирования // ИТНОУ: информационные технологии в науке, образовании и управлении. 2018. № 2. С. 21–26. EDN LBSNXN.
2. Микросервисы и микросервисная архитектура // Atlassian. URL: https://www.atlassian.com/ru/microservices (дата обращения: 20.01.2023).
3. Microservices Use Cases // Alpacked. 2020. 10 August. URL: https://alpacked.io/blog/microservices-use-cases/ (дата обращения: 20.01.2023).
4. *Талипов М.* По данным O'Reilly, микросервисы используют 77 % компаний // Завтра облачно. Журнал VK Cloud об IT-бизнесе, технологиях и цифровой трансформации. 2020. 31 июля. URL: https://mcs.mail.ru/blog/po-dannym-o-reilly-mikroservisy-ispolzuyut-77-kompaniy (дата обращения: 20.01.2023).
5. *Ричардсон К.* Микросервисы. Паттерны разработки и рефакторинга. СПб. : Питер, 2022. 544 с. ISBN 978-5-4461-0996-8.
6. Микросервисная архитектура от А до Я // Axiom JDK. 2021. 2 декабря. URL: https://axiomjdk.ru/announcements/2021/12/02/microservices-101/ (дата обращения: 20.01.2023).
7. Монолитная vs Микросервисная архитектура // Proglib. 2019. 19 сентября. URL: https://proglib.io/p/monolitnaya-vs-mikroservisnaya-arhitektura-2019-09-16 (дата обращения: 20.01.2023).
8. *Perkins K.* Microservices: Breaking it Down // Zirous. 2018. 7 August. URL: https://www.zirous.com/2018/08/07/microservices-breaking-it-down/ (дата обращения: 20.01.2023).

References

1. Oparin G.A., Bogdanova V.G., Pashinin A.A. (2018) Mikroservisy kak fundamentalnaya osnova raspredelennogo sborochnogo programmirovaniya [Microservices as a fundamental basis of distributed aggregate programming]. *Information Technologies in Science, Education and Management.* No. 2. Pp. 21–26. (In Russian).
2. Mikroservisy i mikroservisnaya arkhitektura [Microservices and micro service architecture]. *Atlassian*. URL: https://www.atlassian.com/ru/microservices (accessed 20.01.2023). (In Russian).
3. Microservices Use Cases. *Alpacked*. 2020. 10 August. URL: https://alpacked.io/blog/microservices-use-cases/ (accessed 20.01.2023).
4. Talipov M. (2020) Po dannym O'Reilly, mikroservisy ispolzuyut 77 % kompaniy [Tomorrow is cloudy: 77 % of companies use microservices]. *Zavtra oblachno. VK Cloud journal about IT business, technology and digital transformation.* URL: https://mcs.mail.ru/blog/po-dannym-o-reilly-mikroservisy-ispolzuyut-77-kompaniy (accessed 20.01.2023). (In Russian).
5. Richardson Ch. (2022) *Microservices patterns with examples in Java*. Shelter Island : Manning. (Russian edition: St. Petersburg : Piter Publisher. 544 p. ISBN 978-5-4461-0996-8).
6. *Mikroservisnaya arkhitekturaot A do Ya* [Micro service architecture from A to Z]. *Axiom JDK*. 2021. 2 December. URL: https://axiomjdk.ru/announcements/2021/12/02/microservices-101/ (accessed 20.01.2023). (In Russian).
7. Monolitnaya vs Mikroservisnaya arkhitektura [Monolithic vs Micro Service architecture]. *Proglib*. 2019. 19 September. URL: https://proglib.io/p/monolitnaya-vs-mikroservisnaya-arhitektura-2019-09-16 (accessed 20.01.2023). (In Russian).
8. *Perkins K.* Microservices: Breaking it Down. *Zirous*. 2018. 7 August. URL: https://www.zirous.com/2018/08/07/microservices-breaking-it-down/ (accessed 20.01.2023).